

---

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

---

**Yarin Gal**

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

**Zoubin Ghahramani**

## Abstract

Deep learning tools have recently gained much attention in applied machine learning. However such tools for regression and classification do not allow us to capture model uncertainty. Bayesian models offer us the ability to reason about model uncertainty, but usually come with a prohibitive computational cost.

We show that dropout in multilayer perceptron models (MLPs) can be interpreted as a Bayesian approximation. Results are obtained for modelling uncertainty for dropout MLP models – extracting information that has been thrown away so far, from existing models. This mitigates the problem of representing uncertainty in deep learning without sacrificing computational performance or test accuracy.

We perform an exploratory study of the dropout uncertainty properties. Various network architectures and non-linearities are assessed on tasks of extrapolation, interpolation, and classification. We show that model uncertainty is important for classification tasks using MNIST as an example, and use the model’s uncertainty in a Bayesian pipeline, with deep reinforcement learning as a concrete example.

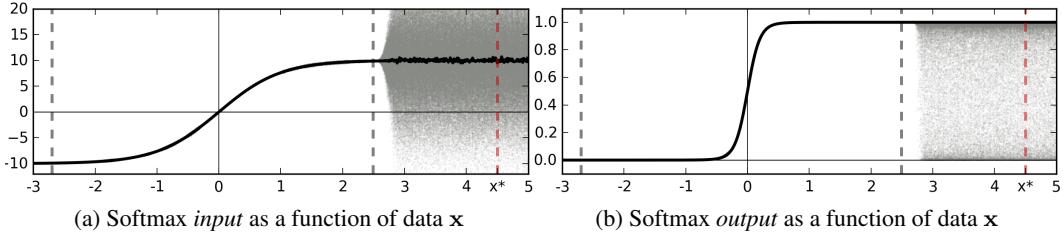
## 1 Introduction

Research in fields such as physics, biology, and manufacturing—to name a few—is using deep learning tools now more than ever [1, 2, 3]. Tools such as the multilayer perceptron (MLP, also known as *neural network*), 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]. Furthermore, there has been a recent shift in many of these fields towards the use of Bayesian uncertainty [6, 7, 8].

Standard deep learning tools for regression and classification do not allow us to 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 about its predictions even with a high softmax output (fig. 1). Passing a point estimate of the mean of a function (solid line 1a) through a softmax (solid line 1b) results in highly confident extrapolations with  $x^*$  (a point far from the training data) 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.

Representing model uncertainty is important for the practical deployment of deep learning systems as well. With model confidence at hand we can take appropriate actions in different scenarios. 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 to classify. 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 have made use of MLPs 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 in this setting, where the agent selects its best action following its current estimation 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 models offer us the ability to reason about model uncertainty, but usually come with a prohibitive computational cost. It is perhaps surprising then that it is possible to use many of the



**Figure 1: A sketch of softmax input and output for an idealised binary classification problem.** Training data is given between the dashed gray 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.

properties often viewed as held by Bayesian models within deep learning without changing either the models or the optimisation. We show that the use of dropout in MLPs can be interpreted as a Bayesian approximation of a well known Bayesian model: the Gaussian process (GP) [12]. Dropout is used in almost all models in deep learning as a way to avoid over-fitting [13]. We develop tools to represent model uncertainty of existing dropout MLP models – extracting information that has been thrown away so far. This mitigates the problem of representing model uncertainty in deep learning without sacrificing either computational performance or test accuracy.

Model uncertainty is often evaluated in exploratory research, studying the properties of the estimated model confidence on different tasks. In this paper we prove the link between the Gaussian process and dropout, and develop the tools necessary to represent uncertainty in deep learning. We then perform an extensive exploratory assessment of the properties of the uncertainty obtained from dropout MLPs and convnets on the tasks of regression and classification. We compare the uncertainty obtained from different model architectures and non-linearities, both on the tasks of interpolation and extrapolation. We show that model uncertainty is important for classification tasks using MNIST as a concrete example. Lastly, we give a quantitative assessment of model uncertainty in the setting of reinforcement learning, on a task similar to that used in deep reinforcement learning [14].

## 2 Related Research

It has long been known that infinitely wide (single hidden layer) MLPs with distributions placed over their weights converge to Gaussian processes [15, 16]. This known relation is through a limit argument that does not allow the transfer of properties from the Gaussian process to finite MLPs. Finite MLPs 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 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 reasonable uncertainty estimates can be cheaply obtained from common dropout models, this results in unnecessary additional computation.

## 3 Dropout as a Bayesian Approximation

We show that a multilayer perceptron (MLP) with arbitrary depth and non-linearities, with dropout applied after every weight layer, is mathematically equivalent to an approximation to the probabilistic deep Gaussian process model [27]. 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. 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 (sections 3). The results are summarised here. In the next section we obtain uncertainty estimates for dropout MLPs.

We denote by  $E$  a loss function such as the softmax loss or the euclidean loss. We denote by  $\mathbf{W}_i$  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  $i = 1, \dots, L$ . We denote by  $\bar{\mathbf{y}}$  the outputs of an MLP model and by  $\mathbf{y}$  the observed outputs

corresponding to inputs  $\mathbf{x}$ . During MLP optimisation, the loss term is scaled by the learning rate  $r_1$  and a regularisation term is added. We often use  $L_2$  regularisation weighted by some weight decay  $r_2$ , resulting in a minimisation objective (often referred to as cost),

$$\mathcal{L}_{\text{dropout}} := r_1 E(\mathbf{y}, \bar{\mathbf{y}}) + r_2 \sum_{i=1}^L (||\mathbf{W}_i||_2^2 + ||\mathbf{b}_i||_2^2). \quad (1)$$

With dropout, we sample binary variables for every input point and for every network unit in each layer. 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 binary variable values in the backward pass propagating the derivatives to the parameters. The probabilities  $p_i$  might be optimised as well.

The deep Gaussian process is a powerful tool in statistics that allows us to model distributions over functions. Assume we are given a covariance function of the form  $\mathbf{K}(\mathbf{x}, \mathbf{y}) = \int p(\mathbf{w})p(b)\sigma(\mathbf{w}^T \mathbf{x} + b)\sigma(\mathbf{w}^T \mathbf{y} + b)d\mathbf{w}db$  with some element-wise non-linearity  $\sigma(\cdot)$  and distributions  $p(\mathbf{w}), p(b)$ . It is straightforward to show that  $\mathbf{K}(\mathbf{x}, \mathbf{y})$  is a valid PSD covariance function – it is an example of a marginalised covariance function [28]. In section 3 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 using the following parametric probabilistic model. Let  $\mathbf{b}_i$  be a  $K_{i-1}$  dimensional *binary* random vector for each layer  $i$ . Write<sup>1</sup>  $\widehat{\mathbf{M}}_i = \mathbf{M}_i \text{diag}(\mathbf{b}_i)$  with matrices  $\mathbf{M}_i$  of dimensions  $K_i \times K_{i-1}$  and let  $\boldsymbol{\omega} = \{\widehat{\mathbf{M}}_i\}_{i=1}^L$ . We use a parametrisation for  $\boldsymbol{\omega}$  which is similar to the well known Gaussian distribution re-parametrisation, where  $x \sim \mathcal{N}(\mu, s^2)$  is written as  $x = \mu + s\epsilon$  with  $\epsilon \sim \mathcal{N}(0, 1)$ . When the Gaussian distributions is used in variational inference,  $\mu$  and  $s$  would be treated as a variational parameter. In our case,  $\mathbf{M}_i$  are treated as variational parameters. The variational distribution,  $q(\boldsymbol{\omega})$ , is defined by

$$\mathbf{b}_{i,j} \sim \text{Bernoulli}(p_i) \text{ for } i = 1, \dots, L, j = 1, \dots, K_{i-1} \quad (2)$$

given some probabilities  $p_i$ . In effect,  $q(\boldsymbol{\omega})$  is a distribution over matrices whose columns can randomly be set to zero. The binary variable  $\mathbf{b}_{i,j} = 0$  indicates that unit  $j$  in layer  $i - 1$  is dropped out as an input to layer  $i$ .

Given bias vectors  $\mathbf{m}_i$  of dimensions  $K_i$  for each layer, the approximate model  $q(\mathbf{y}|\mathbf{x})$  is defined as

$$q(\mathbf{y}|\mathbf{x}) = \int p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega})q(\boldsymbol{\omega})d\boldsymbol{\omega} \quad (3)$$

$$p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega}) = \mathcal{N}(\mathbf{y}; \widehat{\mathbf{y}}(\mathbf{x}; \widehat{\mathbf{M}}_1, \dots, \widehat{\mathbf{M}}_L), \tau^{-1}\mathbf{I}_D) \quad (4)$$

$$\widehat{\mathbf{y}}(\mathbf{x}; \widehat{\mathbf{M}}_1, \dots, \widehat{\mathbf{M}}_L) = \sqrt{\frac{1}{K_L}} \widehat{\mathbf{M}}_L \sigma \left( \dots \sqrt{\frac{1}{K_1}} \widehat{\mathbf{M}}_2 \sigma (\widehat{\mathbf{M}}_1 \mathbf{x} + \mathbf{m}_1) \dots \right) \quad (5)$$

for some precision parameter  $\tau > 0$  and some element-wise non-linearity  $\sigma(\cdot)$ .<sup>2</sup> This approximate model can be derived by placing an approximating 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 (section 3.1 in the appendix).

In variational inference we minimise the KL divergence between the approximate model  $q(\boldsymbol{\omega})$  above and the posterior of the full deep GP –  $p(\mathbf{Y}|\mathbf{X})$ . The minimisation objective is obtained from this KL divergence by Monte Carlo integration over  $\boldsymbol{\omega}$  with a single sample (section 3.4 in the appendix):

$$\mathcal{L}_{\text{GP-MC}} \propto \gamma \tau E(\mathbf{y}, \widehat{\mathbf{y}}) + \gamma \sum_{i=1}^L \left( \frac{p_i}{2} ||\mathbf{M}_i||_2^2 + \frac{1}{2} ||\mathbf{m}_i||_2^2 \right) \quad (6)$$

with  $E(\mathbf{y}, \widehat{\mathbf{y}}) = -\log p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega})$  and  $\mathbf{b}_{i,j}$  realisations from the Bernoulli distribution. This is identical to eq. (1) for an appropriate setting of precision hyper-parameter  $\tau$  and scale parameter  $\gamma$ .

## 4 Obtaining Model Uncertainty

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

---

<sup>1</sup>The  $\text{diag}(\cdot)$  operator maps a vector to a diagonal matrix whose diagonal is the elements of the vector.

<sup>2</sup>We derive the case for regression where  $\mathbf{y} \in \mathbb{R}^D$  but the extension to classification is simple. This is given in section 3.5 in the appendix.

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

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

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  $\{\mathbf{b}_1^t, \dots, \mathbf{b}_L^t\}_{t=1}^T$  with probabilities  $\{p_1, \dots, p_L\}$  and estimate

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

following proposition C in the appendix. We refer to this Monte Carlo estimate as *MC dropout*. In practice this is equivalent to performing  $T$  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 *sensible* 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 predictive uncertainty in the same way:

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

following proposition D in the appendix. In conclusion, to obtain the model's predictive variance we estimate:

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

which equals the sample variance of  $T$  forward passes through the MLP plus the inverse model precision. As the model precision is often embedded in the ratio between the learning rate  $r_1$  and weight decay  $r_2$  in MLPs, one would set  $\tau = \frac{r_1}{r_2}$ .

The predictive distribution  $q(\mathbf{y}^* | \mathbf{x}^*)$  is expected to be highly multi-modal, and the above approximation only gives 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 MLP 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 MLP 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

Model uncertainty is often evaluated in exploratory research, studying the properties of the estimated model confidence on different tasks. We next perform an extensive assessment of the properties of the uncertainty estimates obtained from dropout MLPs and convnets on the tasks of regression and classification. We compare the uncertainty obtained from different model architectures and nonlinearities, both on the tasks of interpolation and extrapolation. We show that model uncertainty is important for classification tasks using MNIST [29] as an example. Lastly, we give 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 MLP 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<sub>2</sub> concentrations dataset derived from in situ air samples collected at Mauna Loa Observatory, Hawaii (referred to as *CO<sub>2</sub>*) [30] to evaluate model extrapolation, and the reconstructed solar irradiance dataset (referred to as *solar*) [31] to assess model

interpolation. The datasets are fairly small, with each dataset consisting of about 200 data points. We centre and normalise both datasets.

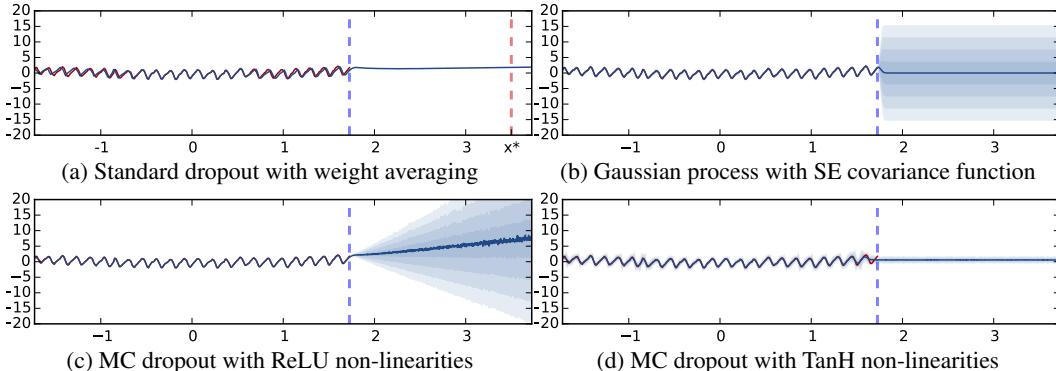
For the task of classification we evaluate the LeNet convolutional neural network model [32] on the MNIST dataset [29]. We show that softmax output probabilities cannot fully capture model uncertainty, and assess model uncertainty in realistic classification cases. Further, we plot a histogram of the samples and assess the moment matching simplifying assumptions above.

### 5.1 Model Uncertainty in Regression Tasks – Extrapolation

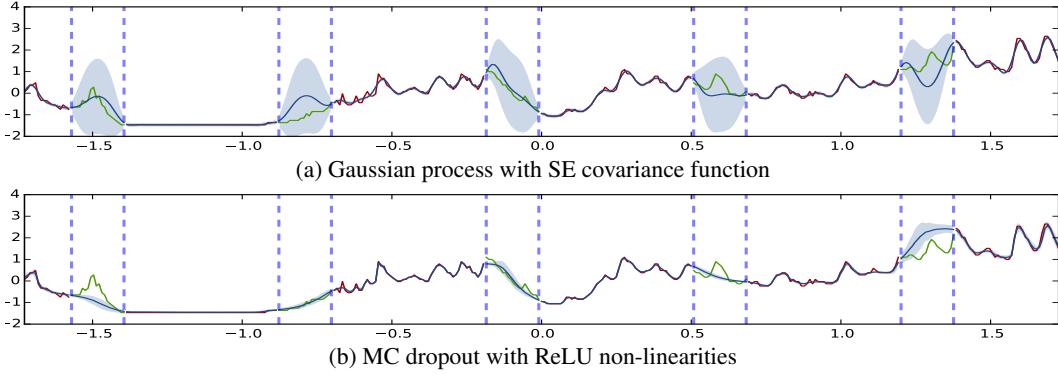
We trained several models on the CO<sub>2</sub> dataset. We use MLPs with either 4 or 5 hidden layers and 1024 hidden units. We use either ReLU non-linearities or TanH non-linearities in each network, and use dropout probabilities of either 0.1 or 0.2. We ran a stochastic gradient descent optimiser for 1,000,000 iterations (until convergence) with learning rate policy  $\text{base-lr} * (1 + \gamma * \text{iter})^{-p}$  with  $\gamma = 0.0001$ ,  $p = 0.25$  and momentum 0.9. We initialise the bias at 0 and initialise the weights uniformly from  $[-\sqrt{3/\text{fan-in}}, \sqrt{3/\text{fan-in}}]$ . We use no mini-batch optimisation as the data is fairly small and with high frequencies. The learning rates used are 0.01 with weight decay of  $1e^{-6}$  for CO<sub>2</sub> and 0.05 with weight decay of  $5e^{-6}$  for solar (corresponding to a high noise precision of  $1e^{-5}$ ). This is to model the low observation noise in the data due to the scaling and high frequencies.

The extrapolation results are shown figure 2. The model is trained on the training data (left to 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 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 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 MLP 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.



**Figure 2: Predictive mean and uncertainties on the Mauna Loa CO<sub>2</sub> 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: dropout confidently predicts an insensible value for the point as the function is periodic; the other models predict an insensible value but with the additional information that the model is uncertain about the prediction.



**Figure 3: Predictive mean and uncertainties on the reconstructed solar irradiance dataset with missing segments, for the GP and MC dropout approximation.** In red is the observed function and in green are the missing segments. In blue is the predictive mean plus/minus two standard deviations of the various approximations.

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 ReLU and TanH approximate different GP covariance functions (section 3.1 in the appendix) and TanH saturates whereas ReLU does not. It is standard in GPs to have different uncertainty estimates for different covariance functions. 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. It is worth mentioning that we attempted to fit the data with models with a smaller number of layers unsuccessfully.

## 5.2 Model Uncertainty in Regression Tasks – Interpolation

For interpolation we repeat the experiment above with ReLU networks with 5 hidden layers and the same setup on a new dataset – solar irradiance. We use base learning rate of  $5e^{-3}$  and weight decay of  $5e^{-7}$ .

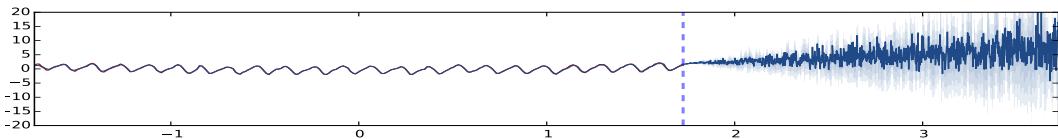
Interpolation results are shown in fig. 3. Fig. 3a shows interpolation of missing sections (bounded between pairs of dashed blue lines) for the Gaussian process with squared exponential covariance function, as well the function value on the training set. In red is the observed function, in green are the missing sections, and in blue is the model predictive mean. Fig. 3b shows the same for the ReLU dropout model with 5 layers.

Both models interpolate the data well, with increased uncertainty over the missing segments. However the GP’s uncertainty is larger and captures the true function within its 2 standard deviation error bars. The observed uncertainty in MC dropout is similar to that of [33]. Note that this is usual with variational techniques, where model uncertainty is often under-estimated.

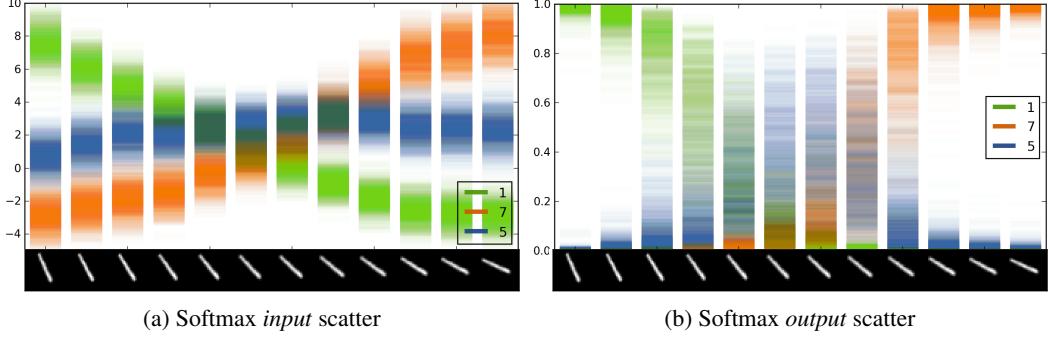
The number of forward iterations used to estimate the uncertainty ( $T$ ) was 1000 for drawing purposes. A much smaller numbers can be used to get a reasonable estimation to the predictive mean and uncertainty (see fig. 4 for example with  $T = 10$ ).

## 5.3 Model Uncertainty in Classification Tasks – MNIST classification

To assess model classification confidence in a real world example we test a convolutional neural network trained on the MNIST dataset. We trained the LeNet convolutional neural network model with dropout applied after the fully connected inner-product layer (after the ReLU operation – the usual way dropout is used in convnets). We used dropout probability of 0.5. We trained the model for 1,000,000 iterations with the same learning rate policy as before with  $\gamma = 0.0001$  and  $p = 0.75$ . We used Caffe [34] reference implementation for this experiment.



**Figure 4: Predictive mean and uncertainties on the Mauna Loa CO<sub>2</sub> concentrations dataset for the MC dropout model with ReLU non-linearities, approximated with 10 samples.**



**Figure 5: 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 extremely large (best viewed in colour).

We evaluated the trained model on a continuously rotated image of the digit 1 (shown on the X axis of fig. 5). We scatter 100 forward passes of the softmax input (the output from the last fully connected layer, fig. 5a), as well as of the softmax output for each class (fig. 5b). 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 the inputs on the X axis. 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 5a), 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.

Note that the histogram shown in fig. 5a is fairly uni-modal, justifying the moment matching suggested in the previous section. This does not need to hold in general though as we only used a single dropout layer in this model. It is interesting to note that the model becomes very confident about the misclassified 7 labels.

#### 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 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 MLPs 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 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 [35] 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. 6). 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.

For the purpose of this experiment, we used future rewards discount of 0.7, no temporal window, and an experience replay of 30,000. The network starts learning after 1,000 steps, where in the initial 5,000 steps random actions are performed. The networks consist of two ReLU hidden layers of size

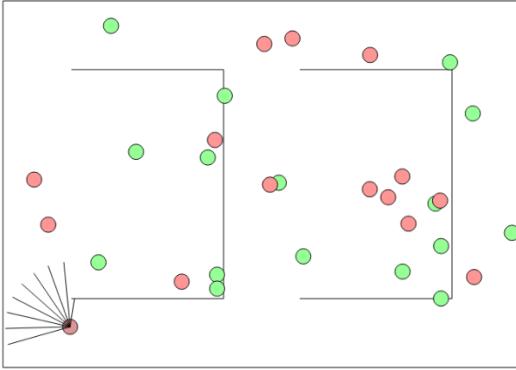


Figure 6: Depiction of the reinforcement learning problem used in the experiments. The agent is in the lower left part of the maze, facing north-west.

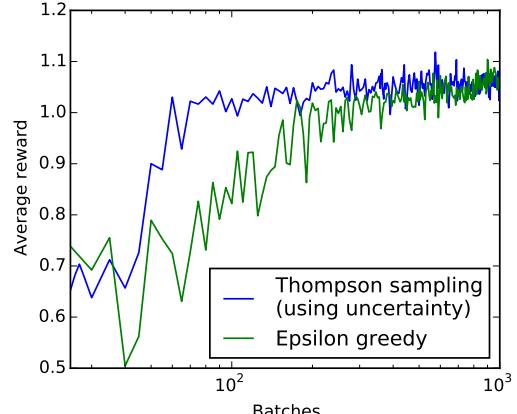


Figure 7: **Log plot of average reward** obtained by both epsilon greedy (in green) and our approach (in blue), as a function of the number of batches.

50, with a learning rate and weight decay of 0.001. Stochastic gradient descent was used with no momentum and batch size of 64.

The original implementation makes use of epsilon greedy exploration with epsilon changing as

$$\epsilon = \min \left( 1, \max \left( \epsilon_{\min}, 1 - \frac{\text{age} - \text{burn-in}}{\text{steps-total} - \text{burn-in}} \right) \right)$$

with steps-total of 200,000, burn-in of 3,000, and  $\epsilon_{\min} = 0.05$ .

We trained the original model and an additional model with dropout with probability 0.1 applied after every non-linearity. 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 forward pass through the network every time we need to make an action. In replay, we perform a single forward pass and then back-propagate with the sampled Bernoulli random variables.

In fig. 7 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 upon 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 of interleaving Bayesian models and deep learning models together with a reinforcement learning example. The developments allows us to treat existing models in a new way without the need to introduce additional parameters or computational burden to obtain uncertainty estimates.

In future research we aim to assess model uncertainty on adversarial inputs, such as corrupted images that classify incorrectly with high confidence [36]. Adding or subtracting a single pixel from each input dimension is perceived as almost unchanged input to a human eye, but can change classification probabilities considerably. In the high dimensional input space the new corrupted image lies far from the data, and one would expect model uncertainty to increase for such inputs.

We compared the dropout uncertainty to that of the Gaussian process model on a variety of tasks on which other GP approximations have been compared as well [33]. In future work we aim to compare the uncertainty of the model to that of Bayesian neural networks such as [26] as well.

## 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, and Mr Yan Wu 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] A Damianou and N Lawrence. Deep Gaussian processes. In *AISTATS*, 2013.
- [28] K Tsuda, T Kin, and K Asai. Marginalized kernels for biological sequences. *Bioinformatics*, 18, 2002.
- [29] Y LeCun and C Cortes. The mnist database of handwritten digits, 1998.
- [30] C D Keeling, T P Whorf, and the Carbon Dioxide Research Group. Atmospheric CO<sub>2</sub> concentrations (ppmv) derived from in situ air samples collected at Mauna Loa Observatory, Hawaii, 2004.
- [31] J Lean. Solar irradiance reconstruction. NOAA/NGDC Paleoclimatology Program, USA, 2004.
- [32] 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.
- [33] Y Gal and R Turner. Improving the Gaussian process sparse spectrum approximation by representing uncertainty in frequency inputs. In *ICML*, 2015.

- [34] 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.
- [35] A Karpathy and authors. A Javascript implementation of neural networks. <https://github.com/karpathy/convnetjs>, 2014–2015.
- [36] C Szegedy, W Zaremba, I Sutskever, J Bruna, D Erhan, I Goodfellow, and R Fergus. Intriguing properties of neural networks. *ICLR*, 2014.