Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning *by Yarin Gal and Zoubin Ghahramani*

ICML 2016

Abtin Mogharabin

Outline

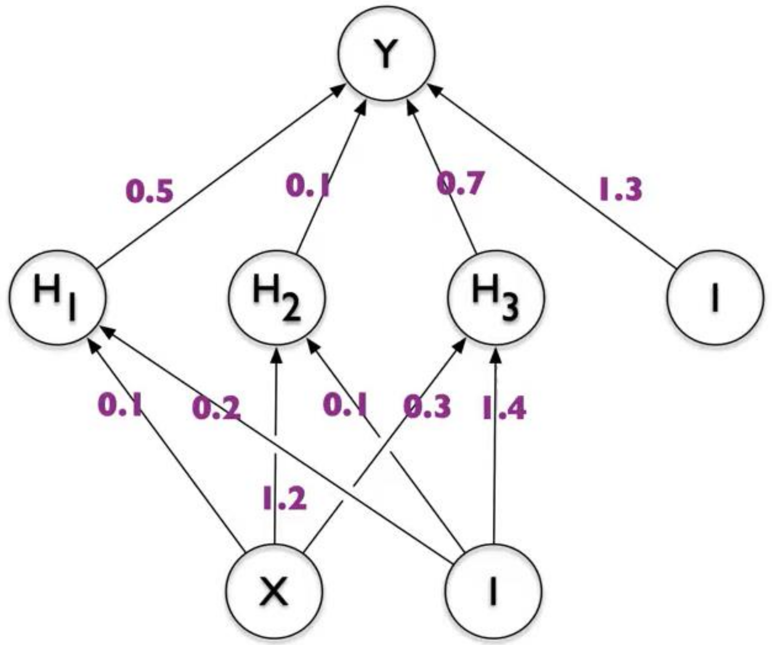
- Introduction
- Methods
- Results
- Relevant Extensions
- Conclusion

Introduction

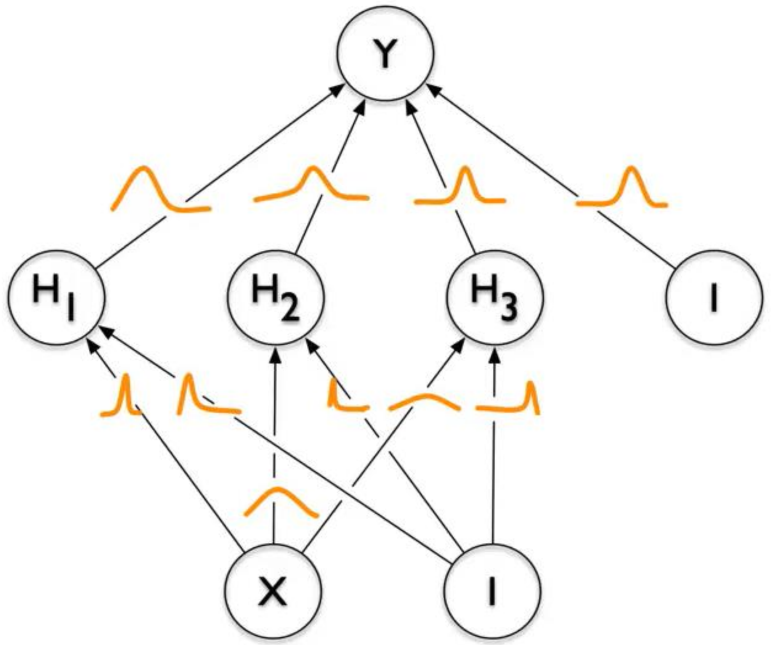
Motivation: Uncertainty-Based Deep Learning

Uncertainty-Based Models:

1. Provides confidence measures along with predictions.
2. Crucial in high-stakes applications (e.g., medical diagnosis, autonomous driving).
3. Separates epistemic (model-related), aleatoric (data-related), and predictive uncertainties.
4. Reduces risk of overconfident predictions.
5. Better handling of out-of-distribution data.

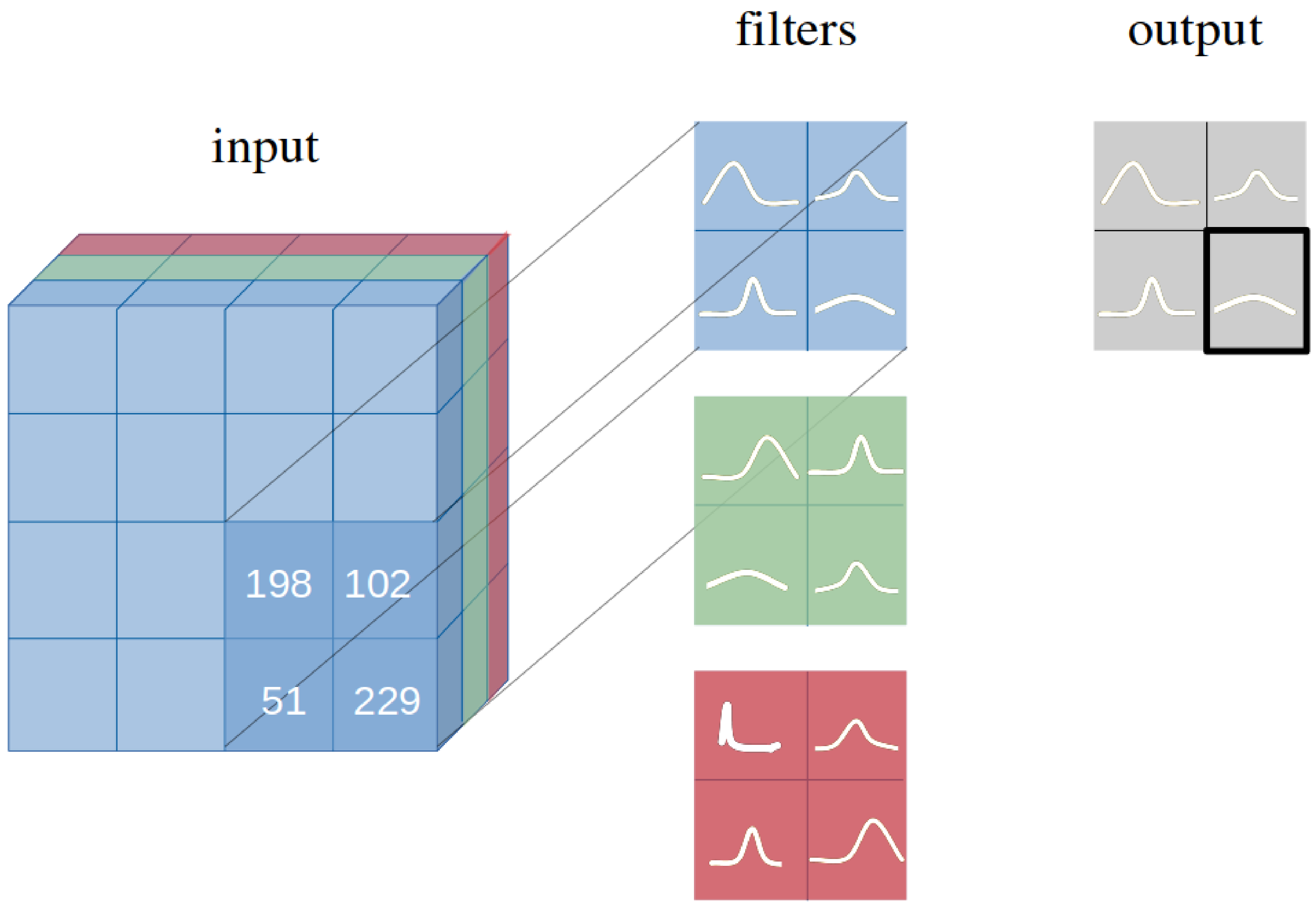


Traditional NN

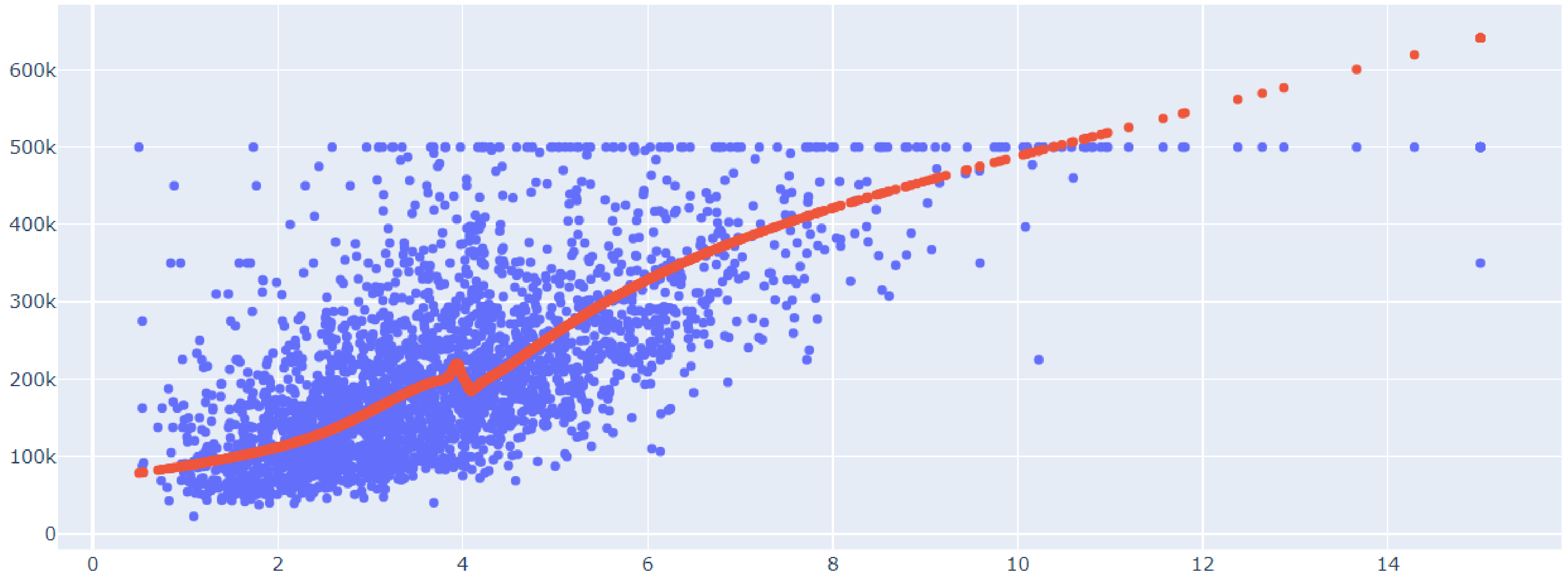


Bayesian NN

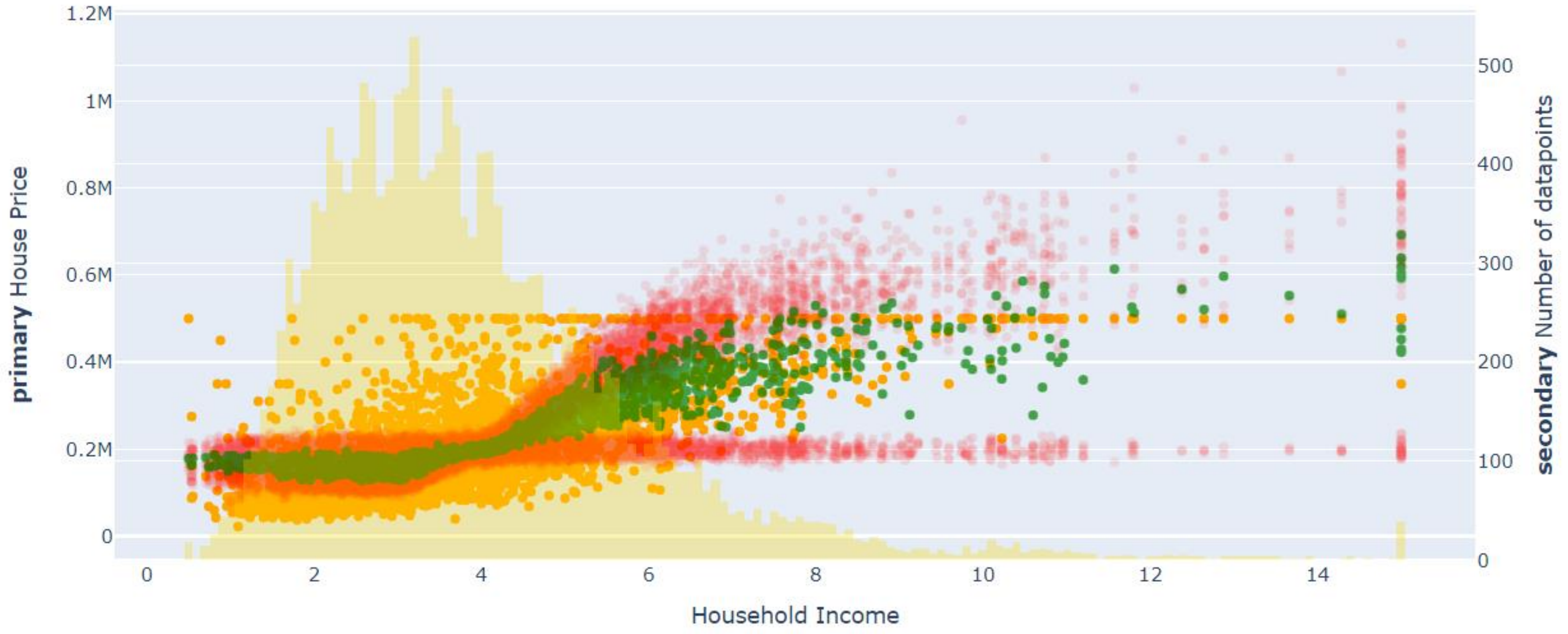
Bayesian Neural Networks



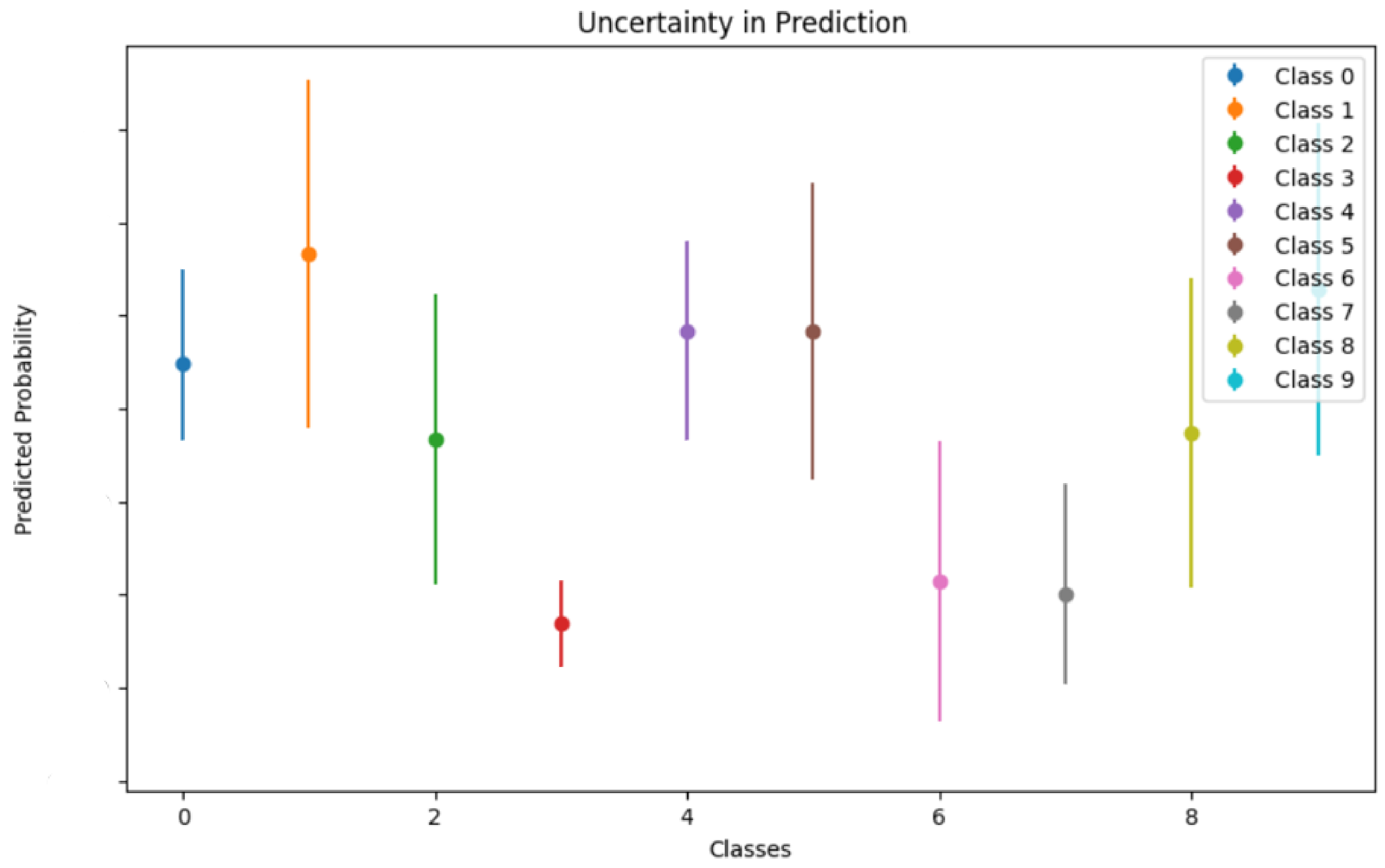
Deterministic Income Prediction



MC-Dropout Income Prediction

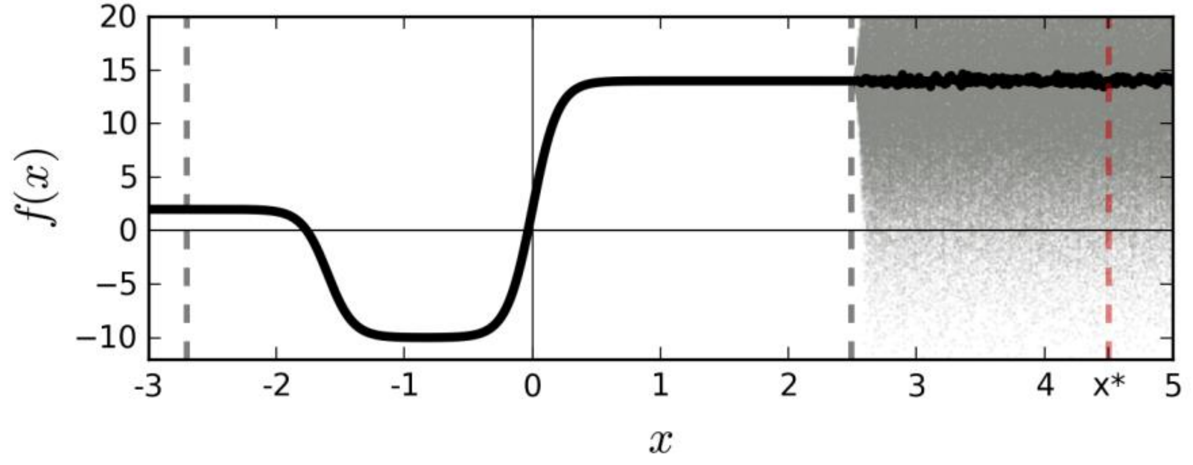


Uncertainty in Classification - CIFAR10

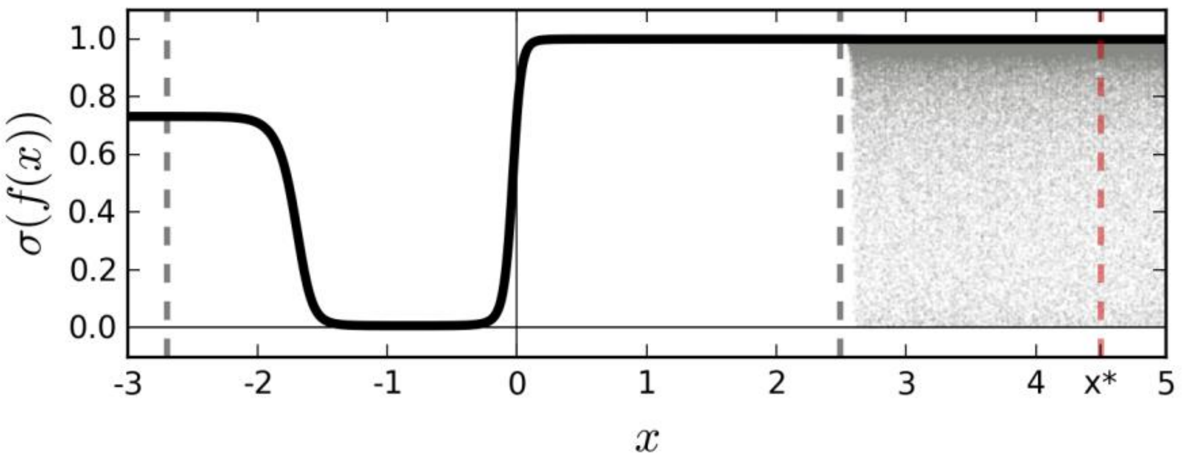


Paper’s Main Contributions

- Paper’s Motivation
  - Deep learning tools are strong but they fail to capture uncertainty
  - Bayesian models offer a computationally expensive but mathematically grounded framework to calculate model uncertainty
- The theory behind dropout
  - Dropout training in NNs is a type of approximate Bayesian inference in GPs
- *Using MC Dropout Approach, we can calculate uncertainty* ***without sacrificing any computational power or train accuracy****!*



(a) Arbitrary function $f(\mathbf{x})$ as a function of data $\mathbf{x}$ (softmax *input*)



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

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.

Dropout as a Bayesian Approximation

The paper shows that dropout is, in fact, just an approximation of variational inference in deep gaussian processes!

Thus, dropout has the potential to provide all the advantages of VI and GPs to us!

Dropout

- In a basic NN with input weight matrix w_1 (Q $* K$), output weight matrix w_2 $(K * D)$, and bias term b (K dimension), the output is:

$$\widehat{\mathbf{y}} = \sigma(\mathbf{x}\mathbf{W}_1 + \mathbf{b})\mathbf{W}_2$$

- In dropout we sample two binary vectors Z_1 (Qdimensions) and Z_2 (K dimensions) such that:

$$\begin{aligned} \mathbf{z}_{1,q} &\sim \text{Bernoulli}(p_1) \text{ for } q = 1, ..., Q \\ \mathbf{z}_{2,k} &\sim \text{Bernoulli}(p_2) \text{ for } k = 1, ..., K \end{aligned}$$

- And the output will be:

$$\widehat{\mathbf{y}} = \sigma(\mathbf{x}(\mathbf{z}_1\mathbf{W}_1) + \mathbf{b})(\mathbf{z}_2\mathbf{W}_2)$$

Gaussian Processes (GPs)

- GPs are a powerful statistical tool that allows us to model distributions over functions in both supervised and unsupervised domains
- They provide:
  - Uncertainty estimates over function values.
  - Robustness to overfitting.
  - Principled hyperparameter tuning.
  - Could be used for large scaled data when cmbined with approximate variational inference

Gaussian Processes (GPs)

- The target is to find the function that generates our data:

$$\mathbf{y} = \mathbf{f}(\mathbf{x})$$

- Using the Bayesian approach, we put a prior over the space of functions p(f)
- We then look for the posterior distribution over the space of functions given our dataset (X, Y):

$$p(\mathbf{f}|\mathbf{X}, \mathbf{Y}) \propto p(\mathbf{Y}|\mathbf{X}, \mathbf{f})p(\mathbf{f})$$

- This distribution captures the most likely functions given our observed data.
- GPs work for both regression and classification tasks

Gaussian Processes (GPs)

- For regression tasks, the prior and the posterior are taken as:

$$\mathbf{F} \mid \mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}(\mathbf{X}, \mathbf{X}))$$

$$\mathbf{Y} \mid \mathbf{F} \sim \mathcal{N}(\mathbf{F}, \tau^{-1}\mathbf{I}_N)$$

- Where τ is the precision hyper-parameter and where I_N is the identity matrix with dimensions N*N.
- K(X, X) is a N × N covariance matrix

Gaussian Processes (GPs)

- For classification tasks, the prior and the posterior are taken as:

$$\mathbf{F} \mid \mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}(\mathbf{X}, \mathbf{X}))$$
$$\mathbf{Y} \mid \mathbf{F} \sim \mathcal{N}(\mathbf{F}, 0 \cdot \mathbf{I}_N)$$

- Here, the precision hyper-parameter is zero.
- Then, we use the output Y values and calculate a categorical distribution with softmax probabilities:

$$c_n \mid \mathbf{Y} \sim \text{Categorical}\left(\exp(y_{nd}) / \left(\sum_{d'} \exp(y_{nd'})\right)\right)$$

Variational Inference (VI)

- VI is a technique used to approximate complex probability distributions in Bayesian models.
- In VI, using basic probability theory rules, we want to find the following predictive distribution that predicts y* for a new input point x*:

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

Variational Inference (VI)

- The main challenge:
  - The distribution p(ω|X, Y) cannot usually be evaluated analytically. Instead we define an approximating variational distribution q(ω)
- Now the training goal is to **get q(ω) as close to p(ω|X, Y) as possible** by minimizing the KL-divergence:

$$\mathrm{KL}(q \parallel p) = \mathbb{E}_q\left[\log \frac{q(z)}{p(z)}\right]$$

Dropout as a Bayesian Approximation

- In a NN model, we minimize a basic E(., .) loss function. And if we add L_2 regularization on the top of it, we'll get:

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

- Depending on the nature of the model, E(., .) could be euclidean loss or softmax loss.
- Here *N* is the number of training examples and *L* is the number of layers in our network.

Dropout as a Bayesian Approximation

- Now we define an approximation of a deep GP model using VI. And show that our minimization task is similar to $\mathcal{L}_{Dropout}$.
- The covariance function K(x, y):

$$\mathbf{K}(\mathbf{x}, \mathbf{y}) = \int p(\mathbf{w})p(b)\sigma(\mathbf{w}^T\mathbf{x} + b)\sigma(\mathbf{w}^T\mathbf{y} + b)\mathbf{d}\mathbf{w}\mathbf{d}b$$

- Here, p(w) and p(b) are the distribution of the weight (w) and bias term (b).
- This covariance function could be approximated by VI.

Dropout as a Bayesian Approximation

- Now take $\omega = \{w_i\}_{\mathrm{i=1}}^{L}$ (for i from 1 to L) with W_i of dimension $K_i * K_{i-1}$ and distributed according to p(w)
- The predictive probability of the deep GP model:

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

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

- And the output $\hat{y}$:

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

Dropout as a Bayesian Approximation

- As we observed before, the posterior $\mathrm{p}(\omega|x, y)$ is our approximation target. Denoted by $\mathrm{q}(\omega)$ and defined as:

$$\mathbf{W}_i = \mathbf{M}_i \cdot \mathrm{diag}([\mathbf{z}_{i,j}]_{j=1}^{K_i})$$
$$\mathbf{z}_{i,j} \sim \mathrm{Bernoulli}(p_i) \text{ for } i = 1, ..., L,\ j = 1, ..., K_{i-1}$$

- Given probabilities p_i and matrices M_i as variational parameters
- We randomly set the columns of M_i to zero

Dropout as a Bayesian Approximation

- Finally, using KL divergence, we minimize:

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

- Target: getting q(ω) as close to p$(\omega|x, y)$ as possible
- Separating the Y and X vectors to a logarithm sum:

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

Dropout as a Bayesian Approximation

- Use Monte Carlo integration with a single sample $\widehat{\omega}_n \sim q(\omega)$ and simplify the result:

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

- Where l is the prior length-scale and τ is the model precision
- Now we set:

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

Putting everything together

- We found the deep GP model using VI minimization target to be:

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

- And the loss of a NN model with dropout and L_2 regularization is:

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

- All left is to take $\tau = pl^2/2N\lambda$ and we'll see: ***dropout is just a Bayesian approximation!***

Obtaining Model Uncertainty

1. First, train a basic deterministic NN model using dropout
2. Then, during the test time, save multiple y predictions for each input x using different (random) dropouts
3. Thus, we have a distribution of $y^*|x^*$
4. Now we observe the approach to calculate the model's uncertainty.

Obtaining Model Uncertainty

- Our approximate predictive distribution is given by

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

- Where $\omega = \{\mathrm{w_i}\}_{\mathrm{i=1}}^{\mathrm{L}}$ is our set of random variables for a model with L layers
- Here, $q(\omega)$ captures the randomness introduced by dropout
- We are interested in the variance (uncertainty) of our prediction y^*

Obtaining Model Uncertainty

- After some not-so-simple calculation, we can calculate:
- The first moment of $q(y^*|x^*)$:

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

- The second moment of $q(y^*|x^*)$:

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

Obtaining Model Uncertainty

- Now, we easily obtain the model's predictive variance (***the model's uncertainty for prediction***):

$$\begin{aligned}\mathrm{Var}_{q(\mathbf{y}^*|\mathbf{x}^*)}\left(\mathbf{y}^*\right) \approx \tau^{-1}\mathbf{I}_D \\ + \frac{1}{T}\sum_{t=1}^{T}\widehat{\mathbf{y}}^*(\mathbf{x}^*, \mathbf{W}_1^t, ..., \mathbf{W}_L^t)^T\widehat{\mathbf{y}}^*(\mathbf{x}^*, \mathbf{W}_1^t, ..., \mathbf{W}_L^t) \\ - \mathbb{E}_{q(\mathbf{y}^*|\mathbf{x}^*)}(\mathbf{y}^*)^T\mathbb{E}_{q(\mathbf{y}^*|\mathbf{x}^*)}(\mathbf{y}^*)\end{aligned}$$

- Remember from a few slides ago that $\tau = pl^2/2N\lambda$

Obtaining Model Uncertainty

- This $Var_{q(y^*|x^*)}(y^*)$ values represents the sample variance of T stochastic forward passes through the NN plus the inverse model precision (τ^{-1})
- For regression, we can also use the predictive log-likelihood:

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

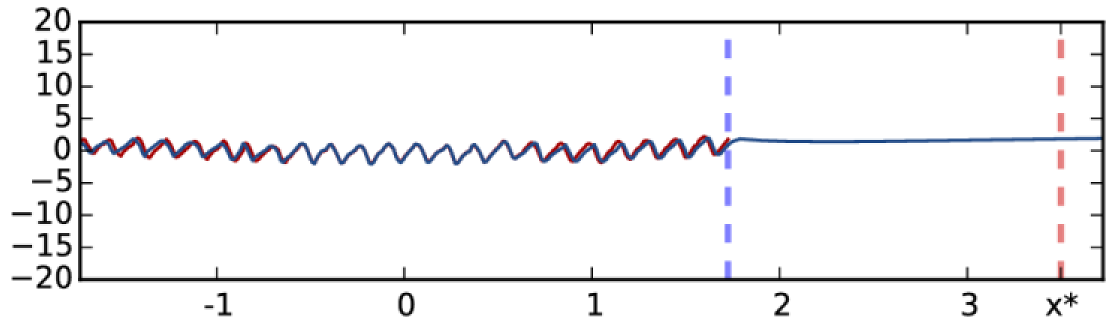
- With a log-sum-exp of T terms and $\hat{y}_t$ stochastic forward passes through the network.

Obtaining Model Uncertainty

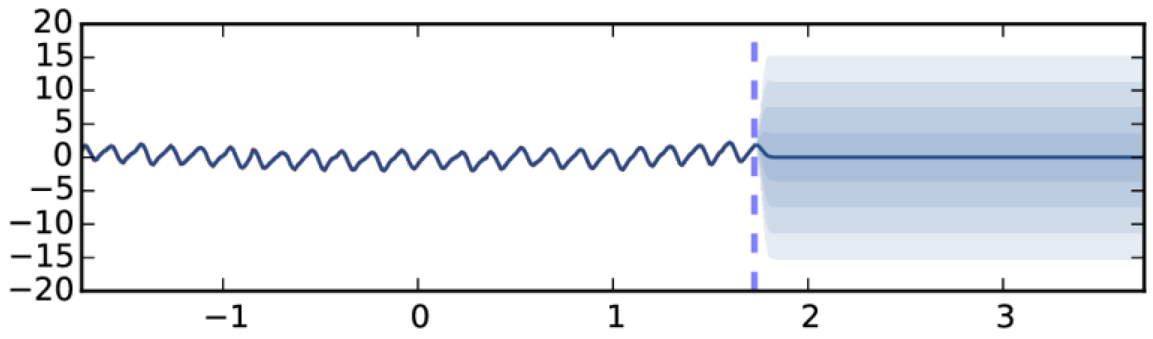
- The variance $Var_{q(y^*|x^*)}(y^*)$:
  - It provides an estimate of the uncertainty in the model's predictions for a given input x^*.
- The predictive log-likelihood $log\ p(y^*|x^*, X, Y)$:
  - It is used to assess how well the model predictions align with actual outcomes, taking into account the training data
- These provide just a glimpse into many properties of our predictive distribution $q(y^*|x^*)$

Results

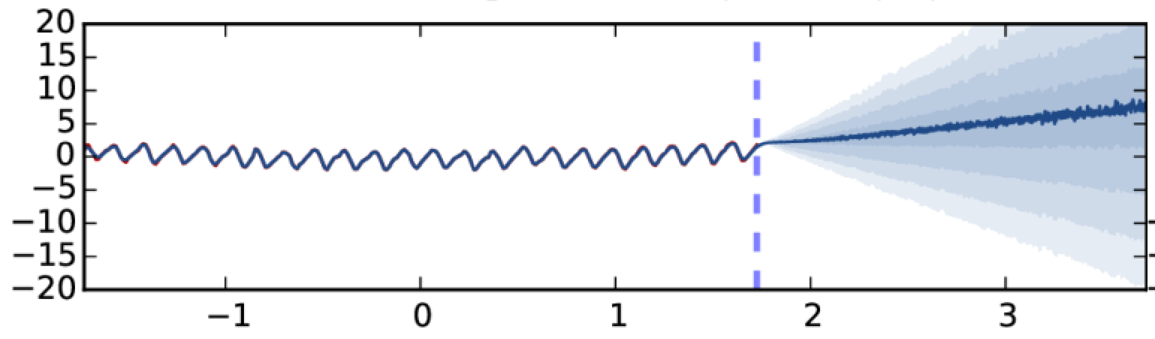
Model Uncertainty in Regression Tasks



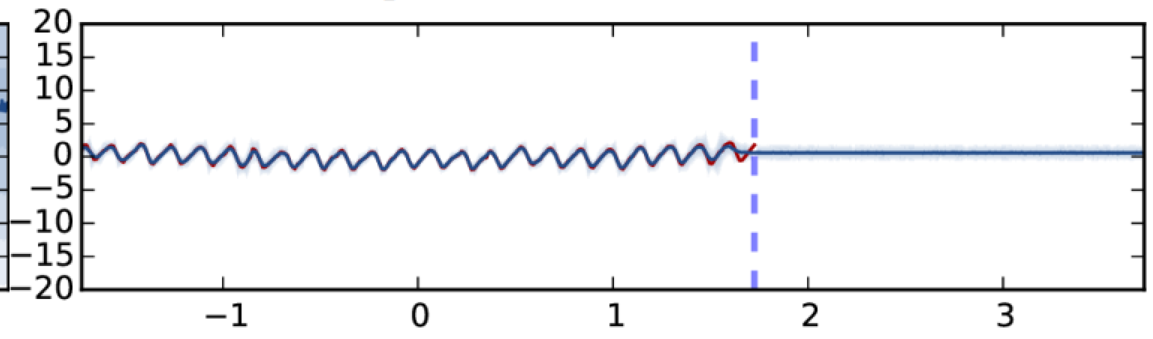
(a) Standard dropout with weight averaging



(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.

Model Uncertainty in Regression Tasks

- Note that the uncertainty is increasing far from the data for the ReLU model, whereas for the TanH model it stays bounded
- This is because dropout’s uncertainty draws its properties from GP and we can prove that:
  - ReLU and TanH approximate different GP covariance functions and TanH saturates whereas ReLU does not.
  - TanH’s saturation means that for large positive or negative inputs, the function's output changes very little, effectively limiting the influence of any further increase or decrease in input.

Model Uncertainty in Regression Tasks

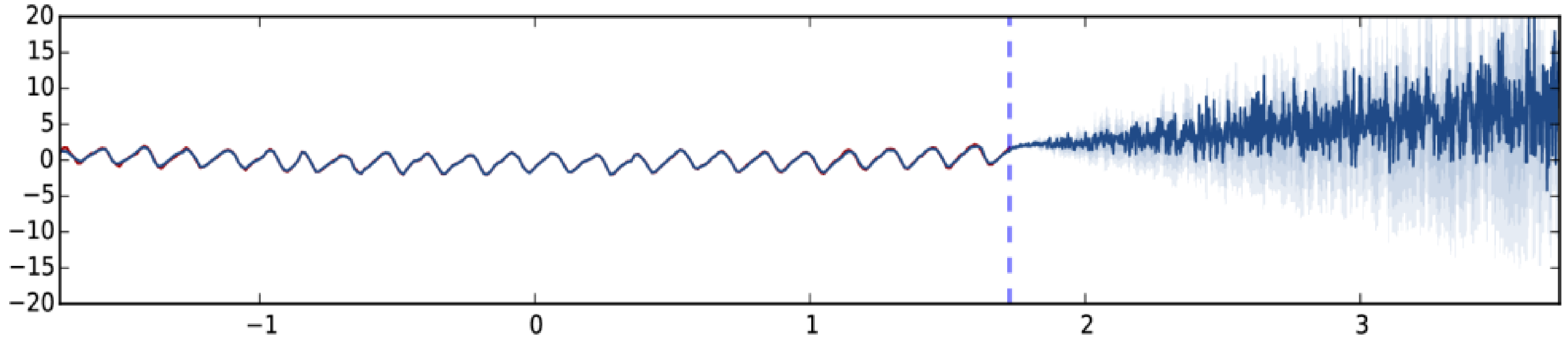
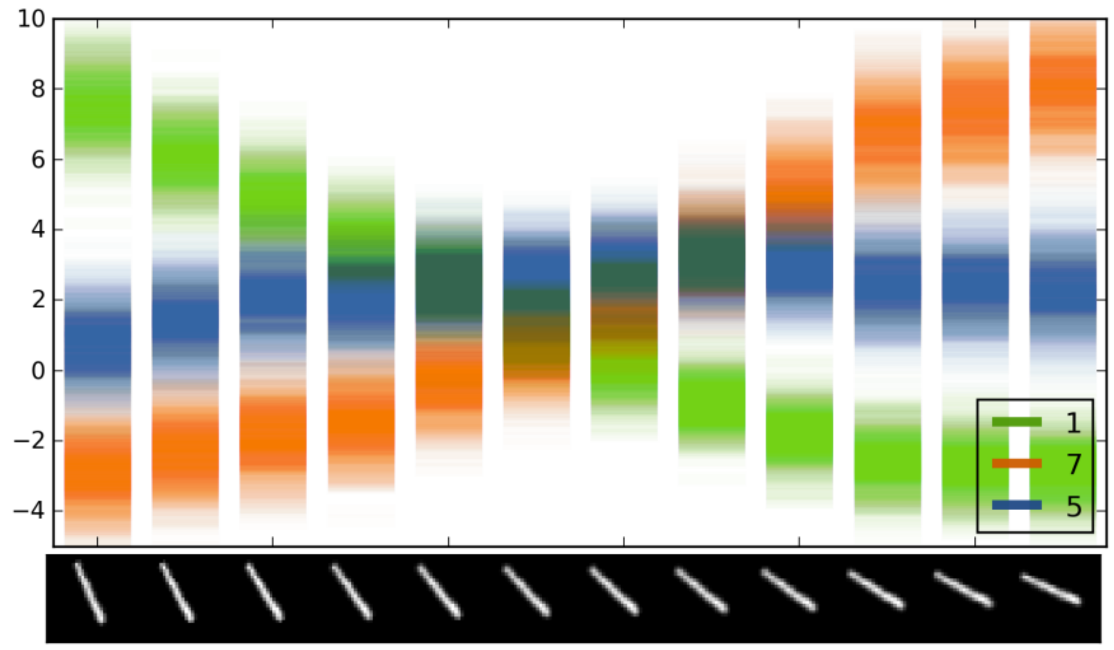


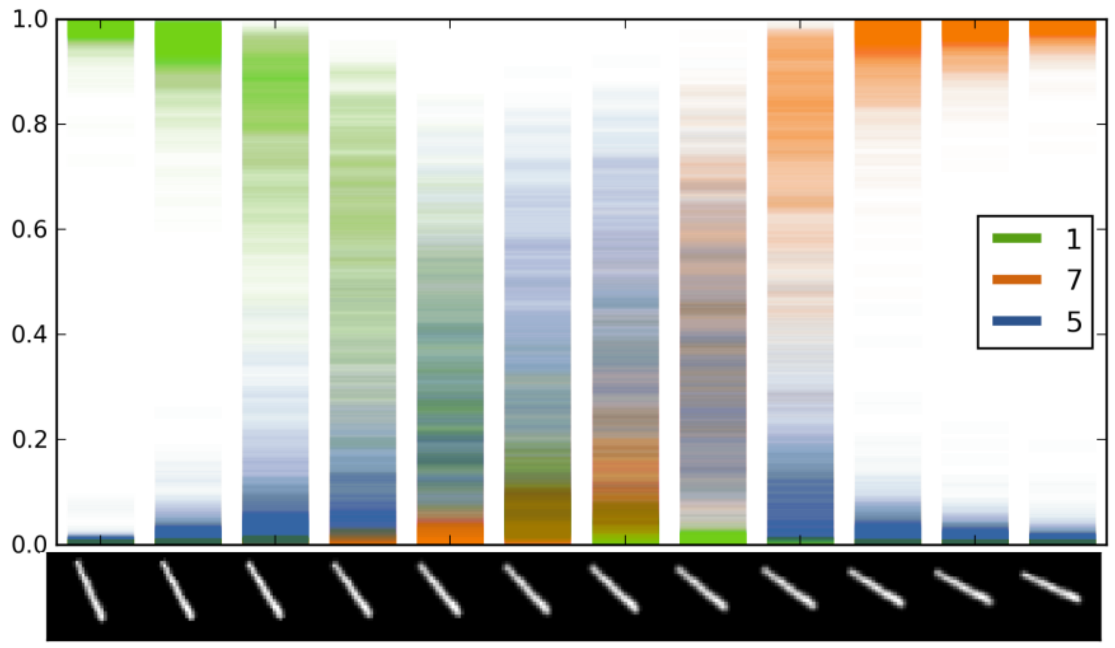
Figure 3. Predictive mean and uncertainties on the Mauna Loa CO_2 concentrations dataset for the MC dropout model with ReLU non-linearities, approximated with 10 samples.

T is taken as 10 here to demonstrate low T effect on uncertainty

Model Uncertainty in Classification Tasks



(a) Softmax *input* scatter

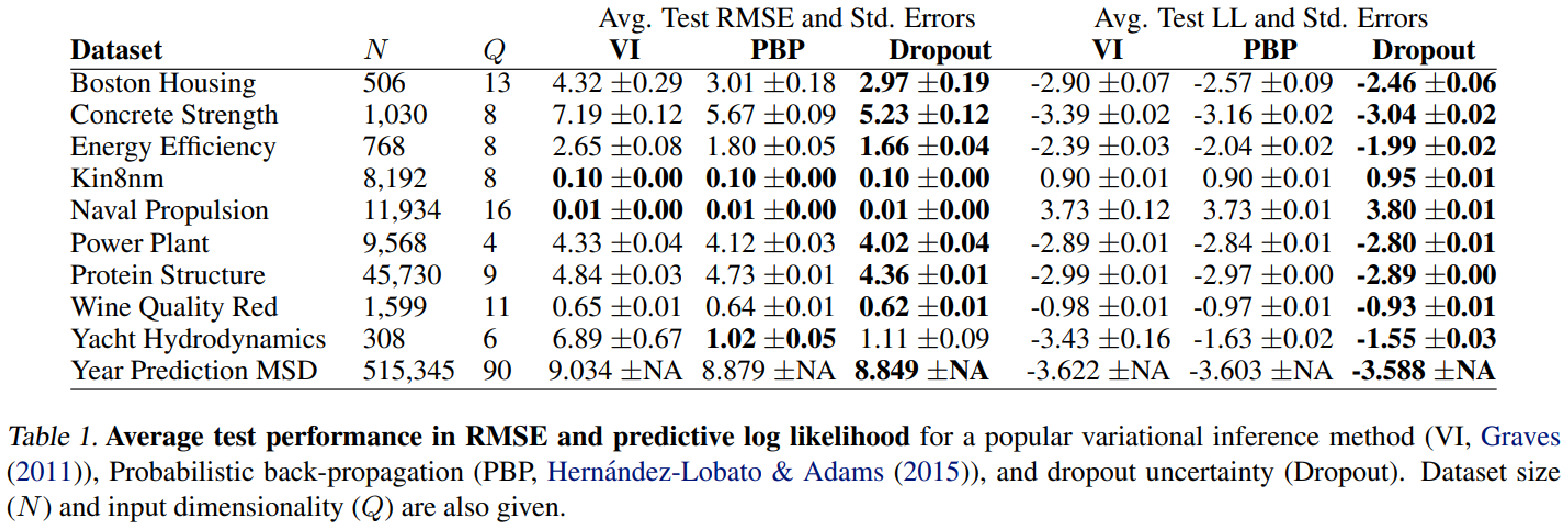


(b) Softmax *output* scatter

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).

Here, they use MNIST dataset for demonstration

Predictive Performance



			Avg. Test RMSE and Std. Errors			Avg. Test LL and Std. Errors		
Dataset	N	Q	**VI**	**PBP**	**Dropout**	**VI**	**PBP**	**Dropout**
Boston Housing	506	13	4.32 ±0.29	3.01 ±0.18	**2.97 ±0.19**	-2.90 ±0.07	-2.57 ±0.09	**-2.46 ±0.06**
Concrete Strength	1,030	8	7.19 ±0.12	5.67 ±0.09	**5.23 ±0.12**	-3.39 ±0.02	-3.16 ±0.02	**-3.04 ±0.02**
Energy Efficiency	768	8	2.65 ±0.08	1.80 ±0.05	**1.66 ±0.04**	-2.39 ±0.03	-2.04 ±0.02	**-1.99 ±0.02**
Kin8nm	8,192	8	**0.10 ±0.00**	**0.10 ±0.00**	**0.10 ±0.00**	0.90 ±0.01	0.90 ±0.01	**0.95 ±0.01**
Naval Propulsion	11,934	16	**0.01 ±0.00**	**0.01 ±0.00**	**0.01 ±0.00**	3.73 ±0.12	3.73 ±0.01	**3.80 ±0.01**
Power Plant	9,568	4	4.33 ±0.04	4.12 ±0.03	**4.02 ±0.04**	-2.89 ±0.01	-2.84 ±0.01	**-2.80 ±0.01**
Protein Structure	45,730	9	4.84 ±0.03	4.73 ±0.01	**4.36 ±0.01**	-2.99 ±0.01	-2.97 ±0.00	**-2.89 ±0.00**
Wine Quality Red	1,599	11	0.65 ±0.01	0.64 ±0.01	**0.62 ±0.01**	-0.98 ±0.01	-0.97 ±0.01	**-0.93 ±0.01**
Yacht Hydrodynamics	308	6	6.89 ±0.67	**1.02 ±0.05**	1.11 ±0.09	-3.43 ±0.16	-1.63 ±0.02	**-1.55 ±0.03**
Year Prediction MSD	515,345	90	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 (2011)), Probabilistic back-propagation (PBP, Hernández-Lobato & Adams (2015)), and dropout uncertainty (Dropout). Dataset size (N) and input dimensionality (Q) are also given.

Model Uncertainty in Reinforcement Learning

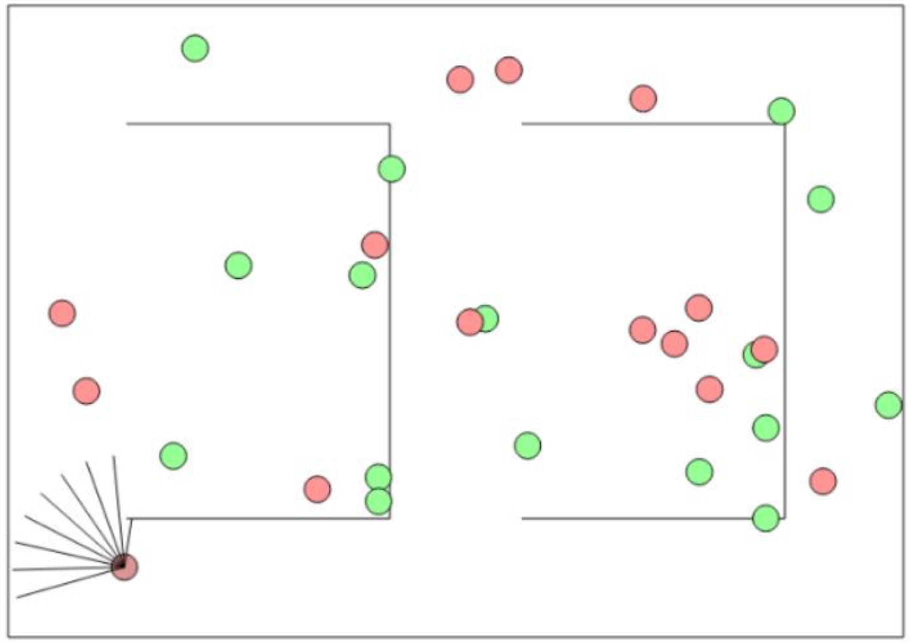


Figure 5. Depiction of the reinforcement learning problem used in the experiments. The agent is in the lower left part of the maze, facing north-west.

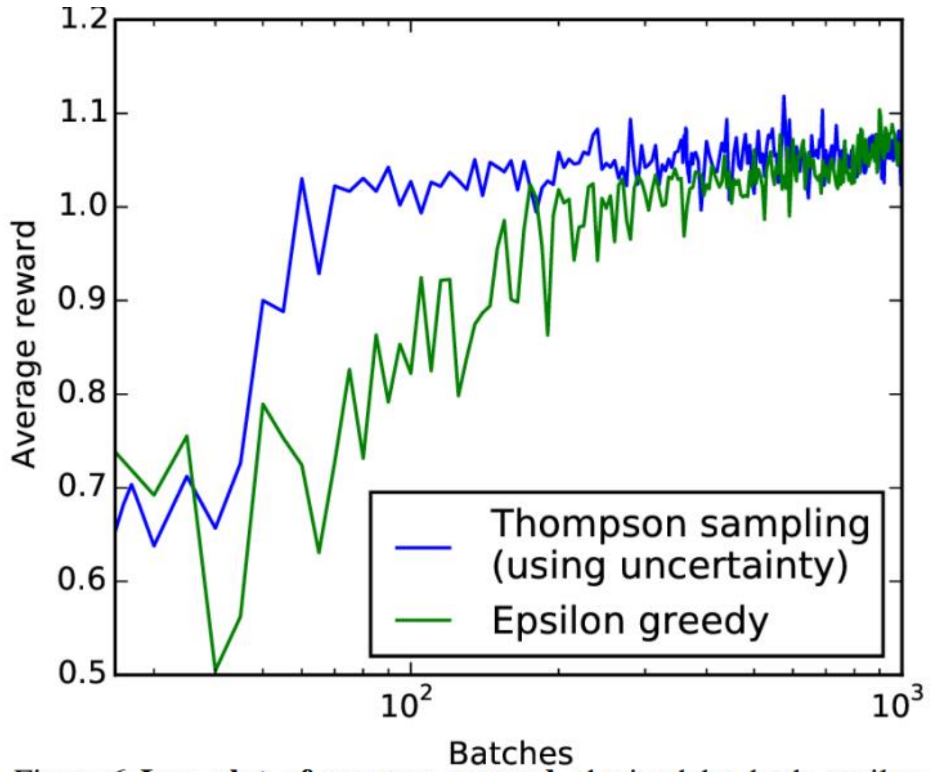


Figure 6. **Log plot of average reward** obtained by both epsilon greedy (in green) and our approach (in blue), as a function of the number of batches.

Model Uncertainty in Reinforcement Learning

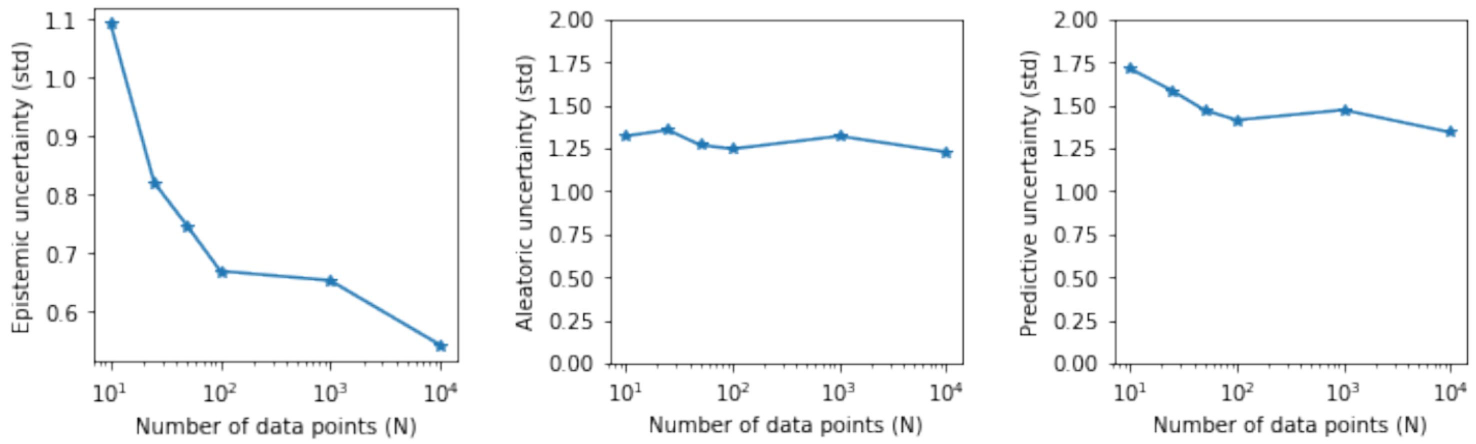
- The log plot of average rewards shows that the Thompson sampling approach using the dropout model converges faster to higher rewards than the epsilon-greedy method.
- Thompson sampling achieved a higher reward threshold within just 25 batches during the initial burn-in period, whereas epsilon-greedy required 175 batches to reach similar performance levels.
- However, it is noted that the performance of the Thompson sampling approach stops improving after 1k epochs.

Relevant Extensions

Concrete Dropout (Gal et al., 2017)

- Dropout rate at each layer is learned as part of the optimization process
- Their approach allows automatic tuning and faster experimentation cycles in large models
- Achieved improved performance and better calibrated uncertainties

Different as the number of data points increases:

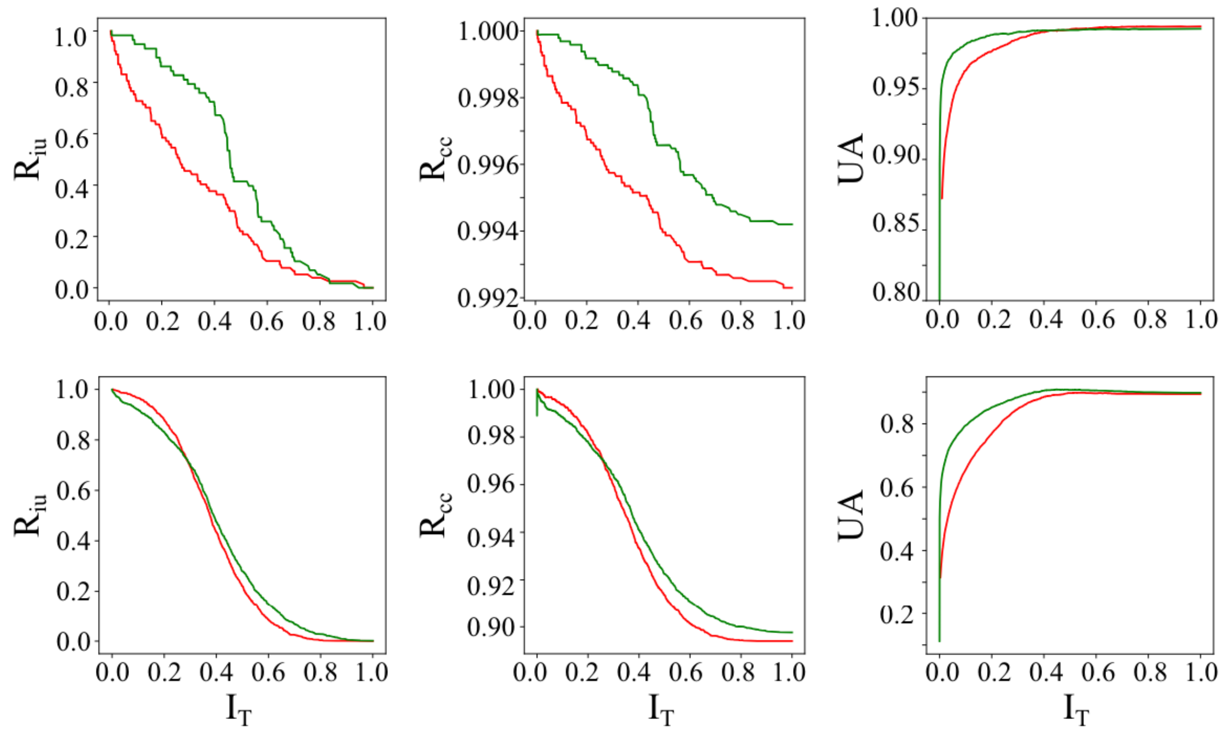


(a) Epistemic (b) Aleatoric (c) Predictive

MC-DropConnect (Mobiny et al., 2021)

- They use the idea of DropConnect from an old 2013 ICML paper (Wan et al.)
- Instead of activations, they set random weights to zero (they put a Bernoulli distribution on model weights)

MC-DropConnect approximated BNN (shown in green) generally performs better than MC-Dropout (shown in red) for both MNIST (Top) and CIFAR-10 (Bottom) datasets:



Conclusion

- We interpret dropout in neural networks as approximate Bayesian inference within deep Gaussian processes.
- We demonstrated how dropout can be employed to extract and utilize uncertainty information from neural networks.
- MC dropout successfully surpassed the state of the art in +10 datasets in 3 different tasks.

References

Gal, Y., Hron, J., & Kendall, A. (2017). Concrete dropout. *Advances in neural information processing systems*, *30*.

Gal, Y., & Ghahramani, Z. (2016, June). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning* (pp. 1050-1059). PMLR.

Mobiny, A., Yuan, P., Moulik, S. K., Garg, N., Wu, C. C., & Van Nguyen, H. (2021). Dropconnect is effective in modeling uncertainty of bayesian deep networks. *Scientific reports*, *11*(1), 5458.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., & Fergus, R. (2013, May). Regularization of neural networks using dropconnect. In *International conference on machine learning* (pp. 1058-1066). PMLR.