Weight Uncertainty in Neural Networks: Reproducibility Challenge

Jeffrey Mak, William Lee and Sam Clarke

Department of Computer Science, University of Oxford

Objectives

Our chosen paper introduces a Bayesian Deep Learning algorithm called Bayes by Backprop. The algorithm learns a probability distribution on the weights of a neural network. The authors claim this has three benefits over learning point estimates of weights:

- It prevents overfitting as well as dropout.
- It outputs both predictions and its uncertainty in those predictions.
- It provides a solution to the exploration vs. exploitation trade-off in RL.

They run three experiments to justify these three benefits. We aimed to replicate these experiments.

Summary of Paper

How do we learn a posterior distribution on weights? Unfortunately, since the functional form of a neural network does not lend itself to integration (even by numerical methods), proper Bayesian inference is intractable.

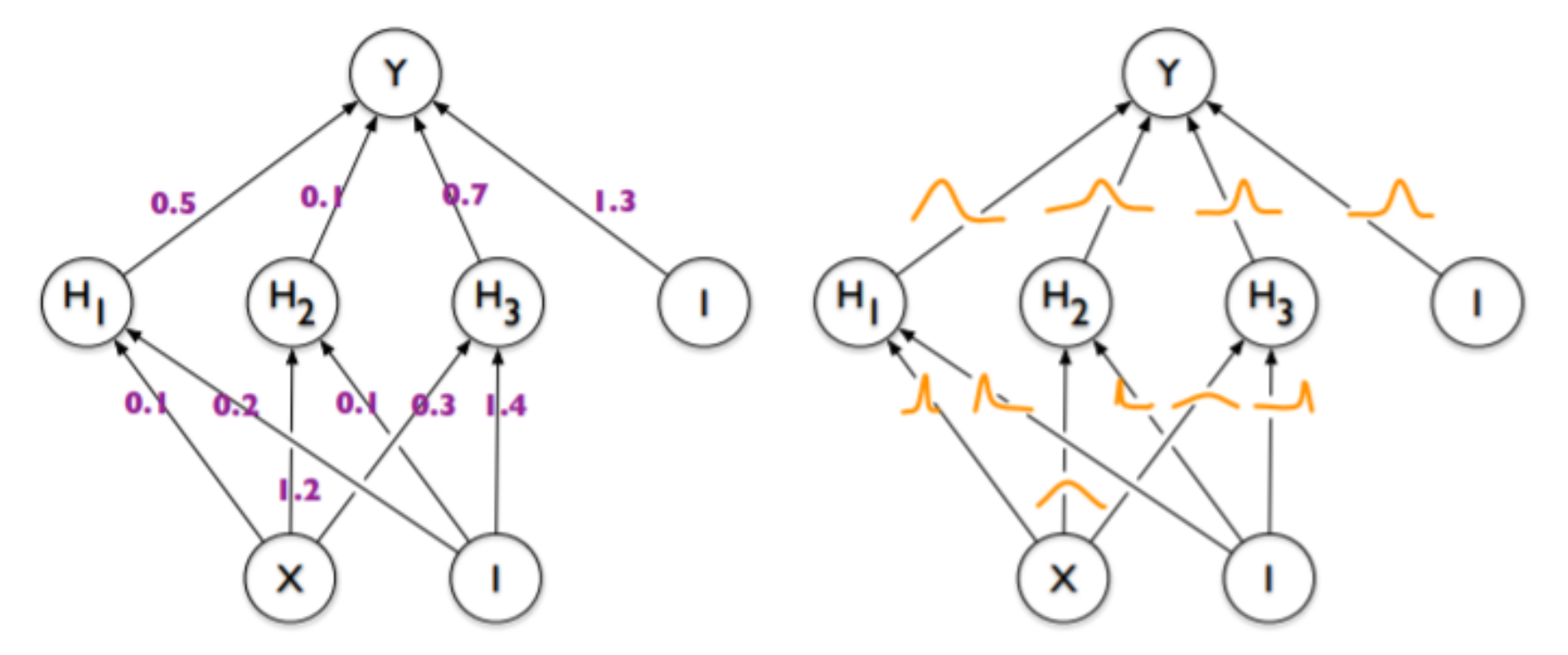

Figure 1: Left: standard learning. Right: being Bayesian.

BBB overcomes this challenge by approximating the true posterior $P(w|D)$ with a variational approximation $q(w|\theta)$ and then optimising θ to minimise $\mathcal{F}(\mathcal{D},\theta)$, the KL divergence with the true posterior. Algorithm 1 implements this by setting (an approximation of) $\mathcal{F}(\mathcal{D},\theta)$ as the loss function then doing standard backpropagation!

$$\begin{aligned}\mathcal{F}(\mathcal{D},\theta) =& \mathrm{KL}\left[q(\mathbf{w}|\theta)||\, P(\mathbf{w}|\mathcal{D})\right]\\ =& \mathbb{E}_{q(\mathbf{w}|\theta)}[\log q(\mathbf{w}|\theta) - \log P(\mathbf{w}) - \\ & \log P(\mathcal{D}|\mathbf{w})]\end{aligned}$$

The Algorithm

Algorithm 1 Bayes by Backprop

```
1:  for all epochs do
2:      $\mathcal{F} \leftarrow 0$
3:      for $i \in \{1..n\}$ do
4:          Sample $\mathbf{w} \sim q(\mathbf{w}|\theta)$
5:          $f = \log q(\mathbf{w}|\theta) - \log P(\mathbf{w}) - \log P(\mathcal{D}|\mathbf{w})$
6:          $\mathcal{F} \leftarrow \mathcal{F} + f$
7:      end for
8:      $\mathcal{F} \leftarrow \mathcal{F}/n$
9:      Backprop $\mathcal{F}$ through model parameters $\theta$
10: end for
```

Experiment 1: MNIST

Does BBB prevent overfitting effectively? To answer this, the paper trains and tests BBB on the MNIST digits dataset. They compare to Dropout, which achieves 1.36% test error. They found BBB to get 1.32% test error. We found BBB to beat dropout by an even bigger margin, as shown in Figure 2.

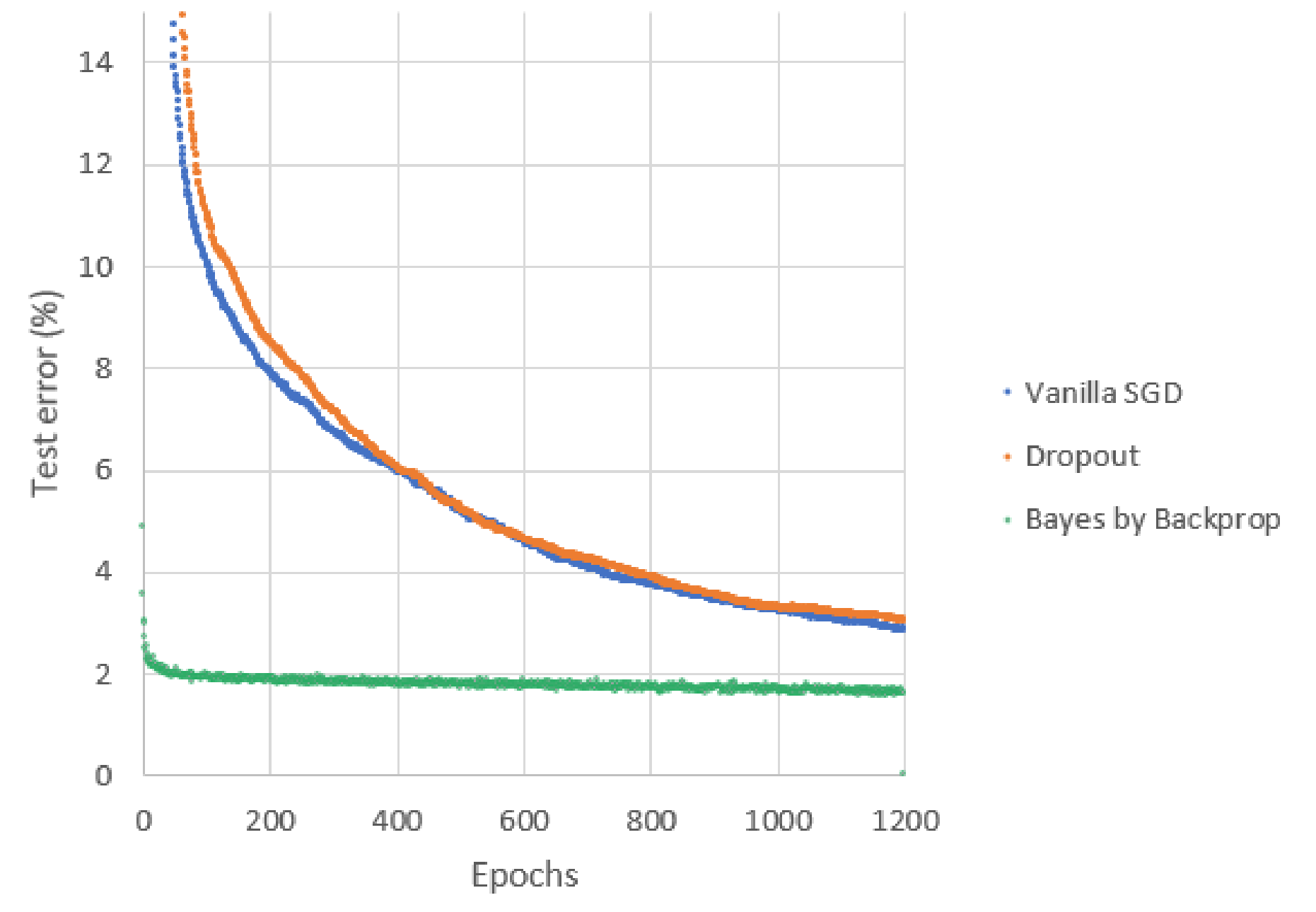

Figure 2: Test error on MNIST as training progresses

Conclusion: Reproduced!

Experiment 2: Regression

Does BBB allow the network to assess its uncertainty in its predictions? The original paper generates training data for a sinusoidal function and compares how networks trained with SGD and BBB fit the data. SGD fits a single function outside the region where there is data. BBB predictions have appropriately high variance. Figure 3 shows our attempt to replicate this result.

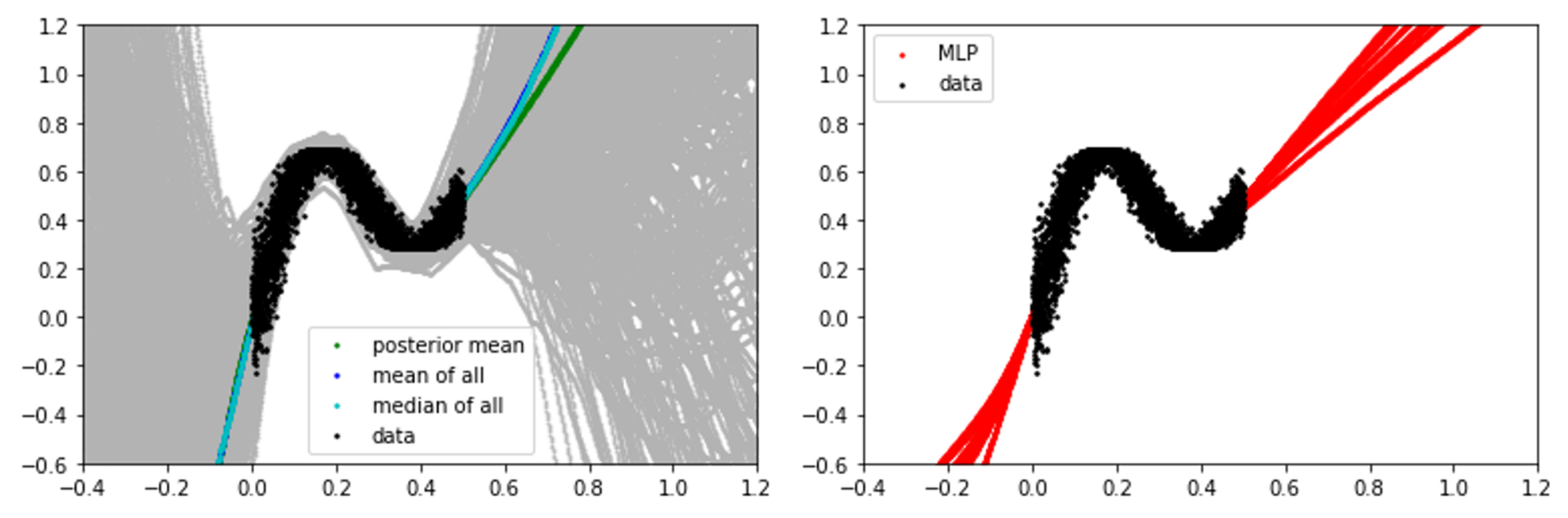

Figure 3: Left: BBB. Right: ensemble of 10 MLPs.

Conclusion: Reproduced, but initialising the weights took at least 24 person-hours!

Experiment 3: Exploration vs. Exploitation

How does an RL agent trained with BBB trade-off exploration vs. exploitation? The authors evaluate its performance on a simple RL problem where the agent must decide whether to eat possibly poisonous mushrooms to maximise its expected cumulative reward. The original paper found BBB to beat simple greedy agents.

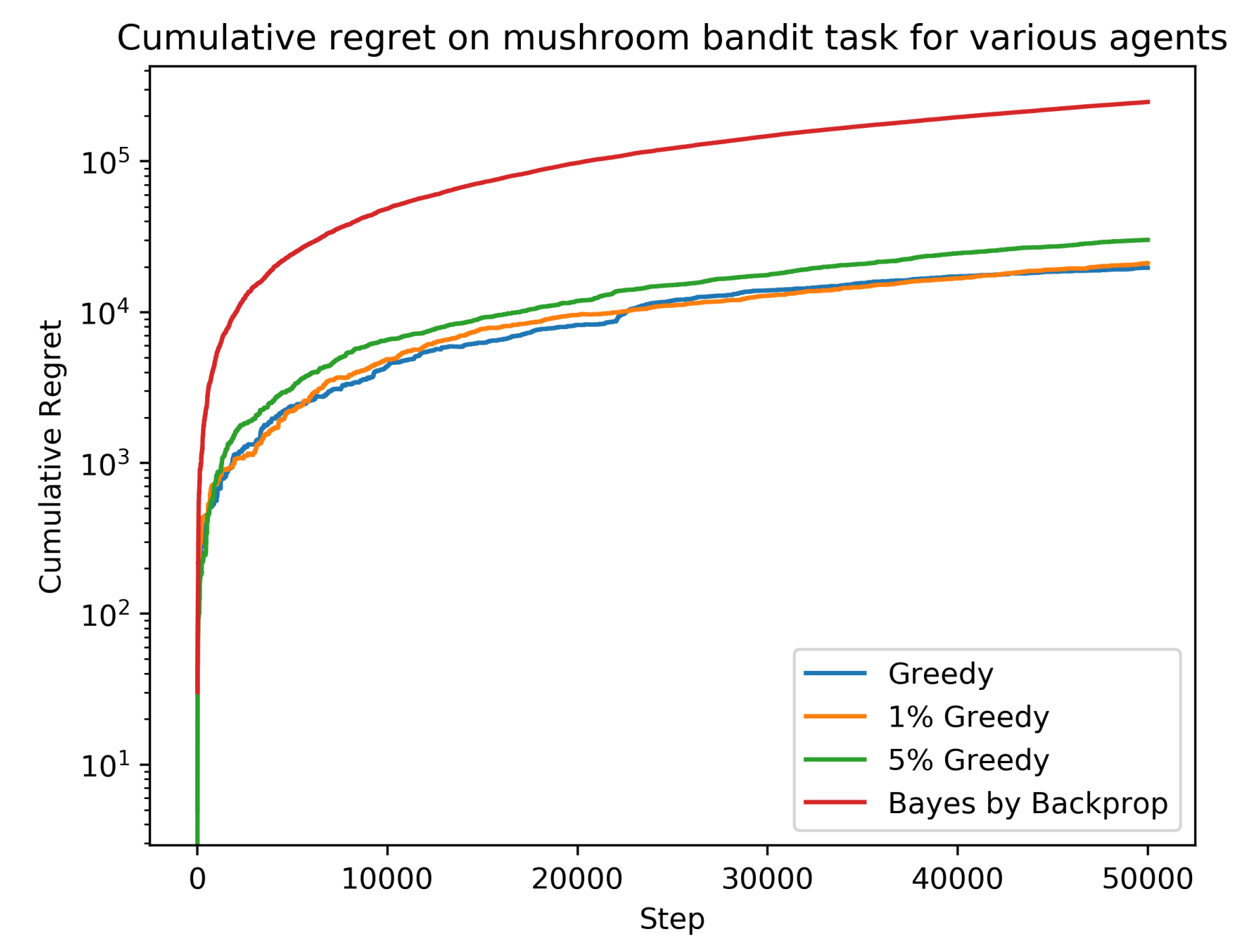

Figure 4: BBB is beaten by simple greedy agents.

Conclusion: Not reproduced.

Extension 1: BBB v. Dropout

In light of our struggles with hyperparameter tuning, we compared BBB with another method of Bayesian approximation for neural networks: Dropout [2]. On the regression task, it gave similar results with no hyperparameter tuning and 20x less training.

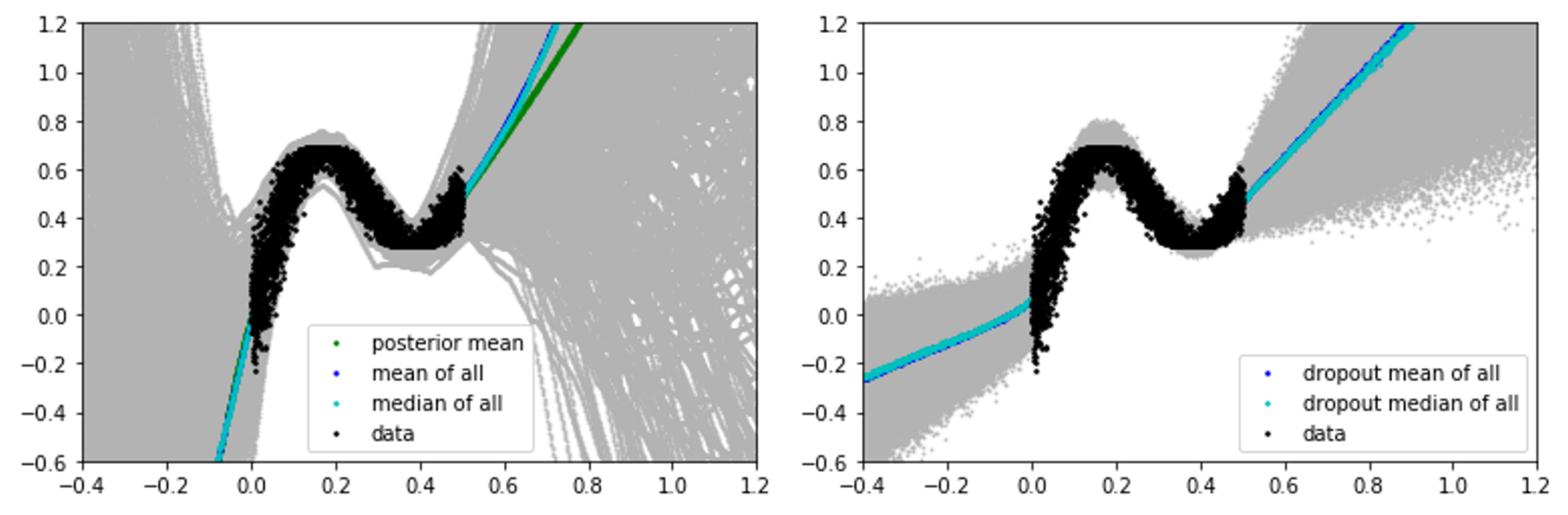

Figure 5: Dropout is way better.

Extension 2: Active Learning

Can BBB be used to request labels with the highest value of information in an active learning setting?

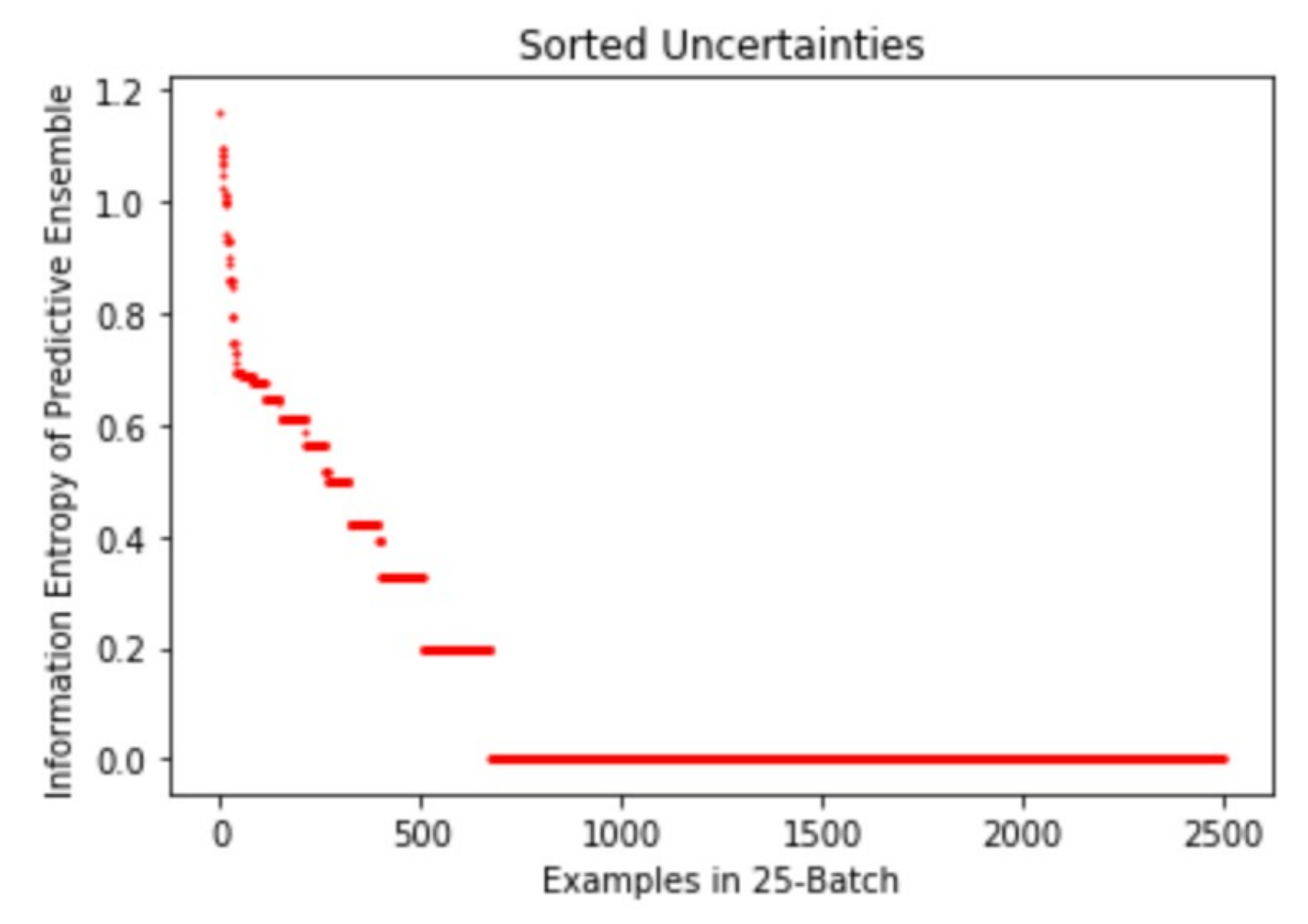

Figure 6: Ordering MNIST examples by information entropy.

Conclusion

1. 2/3 experiments replicated.
2. BBB requires costly hyperparameter tuning to be as cool as it seemed to be.

References

[1] Blundell, C., et al. (2015). Weight uncertainty in neural networks.

[2] Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation.