MC Dropout: Reproducing Model Uncertainty Experiments from Gal & Ghahramani (2016)

Zabdy Leos

Abstract

In this report I reproduce and analyze the key experiments presented in the paper *"Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning"* by Gal and Ghahramani (2016). The work explores how dropout can be interpreted as a form of approximate Bayesian inference, and how Monte Carlo Dropout (MC Dropout) enables uncertainty estimation in deep learning. Both regression and classification experiments were implemented in PyTorch, following the original methodology. Results confirm the ability of MC Dropout to capture model uncertainty effectively, especially in out-of-distribution regions, aligning with the original paper's conclusions.

1 Introduction

Deep neural networks often lack mechanisms to estimate uncertainty in their predictions, which is crucial for real world applications like medical diagnosis, robotics, or autonomous driving. Dropout, originally introduced as a regularization technique, was reinterpreted by Gal and Ghahramani (2016) as approximate Bayesian inference in deep Gaussian processes. This approach, known as Monte Carlo Dropout (MC Dropout), allows for scalable uncertainty estimation by keeping dropout active at test time and averaging multiple stochastic predictions.

This report reproduces the original paper's key experiments in both regression and classification using PyTorch. We analyze how different network configurations express uncertainty, and compare standard dropout inference to MC sampling, visualizing model confidence and its limits.

The full implementation is available at:`https://github.com/zleoslun/dropout-bayes.git`

2 Background: Bayesian View of Dropout

Bayesian inference in neural networks seeks to model uncertainty by placing a distribution over weights. The predictive distribution becomes:

$$p(y^*|\mathbf{x}^*, \mathcal{D}) = \int p(y^*|\mathbf{x}^*, W)\, p(W|\mathcal{D})\, dW$$

This is intractable, so Gal and Ghahramani (2016) proposed using dropout as a variational approximation. With dropout and L_2 regularization, training minimizes:

$$\mathcal{L} = \frac{1}{N} \sum_i \|y_i - \hat{y}_i\|^2 + \lambda \sum_l \|W_l\|^2$$

To estimate uncertainty, dropout remains active at test time. Running T stochastic passes yields predictive mean and variance:

$$\mathbb{E}[y^*] \approx \frac{1}{T} \sum_{t=1}^{T} \hat{y}^{(t)}, \quad \text{Var}[y^*] \approx \frac{1}{T} \sum_{t=1}^{T} (\hat{y}^{(t)})^2 - (\mathbb{E}[y^*])^2$$

This is known as Monte Carlo Dropout (MC Dropout).

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

2.1 Use Cases

MC Dropout is useful for:

- **Extrapolation:** e.g., in time-series regression.
- **Out-of-distribution detection:** via high predictive variance.
- **Decision-making:** e.g., reinforcement learning, active learning.

3 Experiments

In this section I reproduce the main experiments from Gal & Ghahramani (2016), focusing on predictive uncertainty in both regression and classification. I implemented all models and evaluations in PyTorch, with manual control of dropout behavior during training and inference.

3.1 Regression: Mauna Loa CO_2

I used the deseasonalized Mauna Loa CO_2 dataset, selecting the first 200 points and normalizing them to the $[0, 1]$ interval. All models share the same architecture: three fully connected layers with 100 hidden units, dropout rate $p = 0.1$, trained using MSE loss and L_2 regularization. The goal was to compare predictive mean and uncertainty using standard dropout versus MC Dropout.

Models:

- **Standard Dropout:** Dropout enabled only during training, disabled with `model.eval()` at test time (weight averaging).
- **MC Dropout with ReLU:** Dropout remains active also during inference using `model.train()`, with $T = 100$ stochastic forward passes.
- **MC Dropout with Tanh:** Same as above, but with Tanh activation.

In the implementation, I computed the predictive mean and variance by averaging T predictions with dropout enabled. I also created three variations of the MC Dropout experiment: one limited to the training domain, one extended beyond it ($x \in [0, 2]$), and one with only $T = 10$ samples, to analyze the effect of fewer passes.

3.2 Classification: Rotated MNIST

To test model uncertainty in classification, I trained a LeNet-style model with dropout on MNIST. After training, I selected digit "1" images and rotated them from 0° to 180° in steps of 15°. For each rotated image, I performed $T = 100$ forward passes with dropout enabled.

Implementation: I used batched inputs and recorded softmax logits and probabilities for each rotation angle. I visualized logit scatter plots per angle, probability scatter plots, and class heatmaps similar to Figure 4 of the original paper. This allowed me to observe how prediction entropy increased with rotation, reflecting greater epistemic uncertainty as the image becomes less familiar.

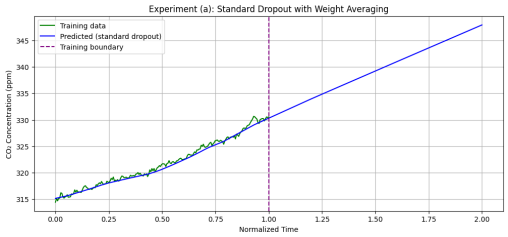
3.3 Comparison with Standard Dropout

I also compared predictions using `model.eval()` (standard dropout) versus MC Dropout. I computed the predicted class for each rotation using both approaches. The model in `eval()` mode is overconfident on heavily rotated digits. MC Dropout, in contrast, produces smoother and less confident predictions, which more accurately reflect model uncertainty in unfamiliar regions.

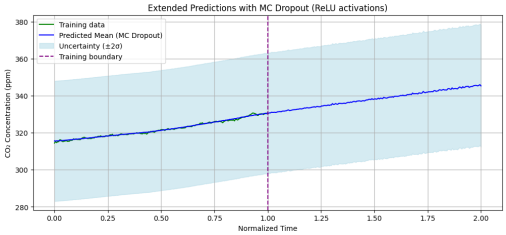
4 Results and Discussion

4.1 Regression: Uncertainty Beyond Training Range

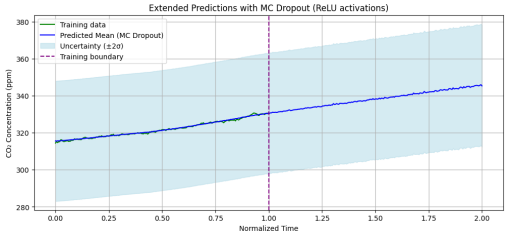
In the Mauna Loa CO_2 experiments (Figure 1), I observed how different configurations express uncertainty when extrapolating beyond the training data. The standard dropout model (Figure 1a)



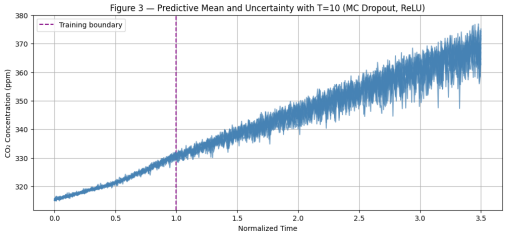
(a) Standard Dropout: Prediction with dropout disabled at test time.



(b) MC Dropout (ReLU) within training domain.

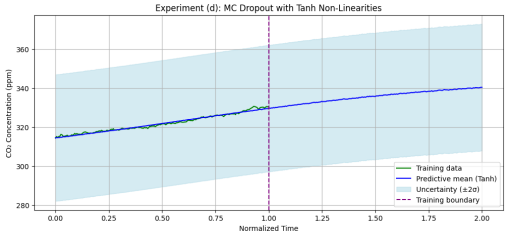


(c) MC Dropout (ReLU) with extrapolation beyond training domain.

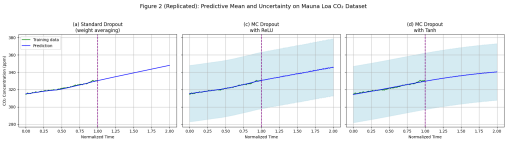


(d) MC Dropout (ReLU) with only $T = 10$ samples.

Figure 1: Regression experiments on Mauna Loa CO_2 data comparing different dropout strategies and their uncertainty estimates.

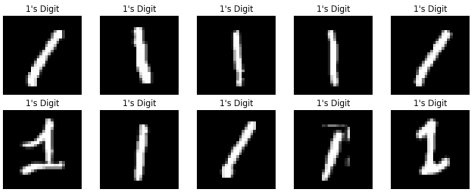


(a) MC Dropout with Tanh activations showing smoother uncertainty bands.

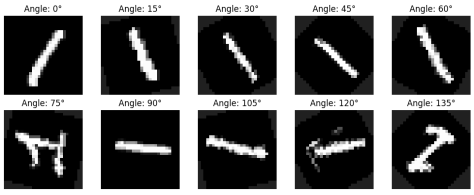


(b) Side-by-side comparison of the three main experiments.

Figure 2: Additional regression results showing the effect of activation functions and comparative analysis.



(a) Samples of digit "1" from MNIST dataset.



(b) Digit "1" rotated from $0°$ to $135°$.

Figure 3: MNIST digit "1" samples and their rotated versions used for uncertainty analysis.

predicts a smooth trend, but completely lacks any representation of uncertainty. This is expected, as dropout is disabled at test time and only a single deterministic forward pass is used.

In contrast, MC Dropout explicitly models predictive variance. In Figure 1b, where inference is performed only on the training domain, the uncertainty remains small but present. However, when I extended the input range to $x \in [0, 2]$ (Figure 1c), the variance increased significantly after $x = 1.0$ — exactly as expected from a Bayesian model, where uncertainty grows in out-of-distribution regions.

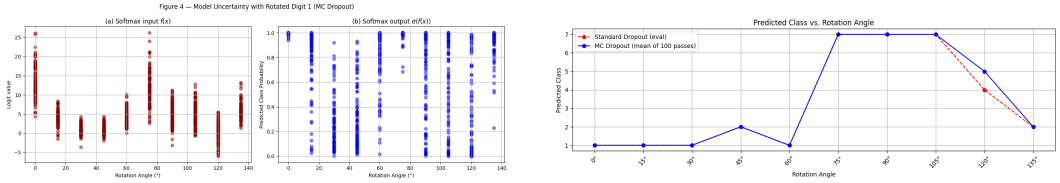
To further validate the stochastic estimation, I reproduced the experiment using only $T = 10$ forward passes (Figure 1d). While the predictive mean remains close to previous runs, the uncertainty appears more noisy — highlighting the tradeoff between sample size and stability in MC Dropout.

Finally, Figure 2a shows results using Tanh non-linearities. The uncertainty bands here are smoother and more calibrated than those with ReLU. This matches the findings of Gal & Ghahramani that Tanh yields more stable approximations, and is useful in applications where gradual change in confidence is desired.

4.2 Classification: Uncertainty on Rotated Inputs

To evaluate how model uncertainty behaves on distorted inputs, I implemented the rotated MNIST experiment proposed in Gal & Ghahramani (2016). I first extracted images of digit "1" from the MNIST training set, as shown in Figure 3a, and then applied rotations from $0°$ to $135°$ in increments of $15°$ (Figure 3b). This rotation simulates increasing deviation from the training distribution.

After training a LeNet-style CNN with dropout ($p = 0.5$), I enabled dropout at test time by calling `model.train()` instead of `model.eval()` — ensuring randomness remained active during inference. I ran $T = 100$ forward passes for each rotated image to obtain distributions of logits and softmax outputs.



(a) Model uncertainty across rotated digits showing logit distributions and predicted probabilities.

(b) Comparison of Standard Dropout vs. MC Dropout predictions.

Figure 4: Classification results showing uncertainty quantification and method comparison.

Figure 4a shows that as rotation increases, the logits become more dispersed and the predicted probabilities show higher entropy — clear indicators of epistemic uncertainty. To assess the benefits of MC Dropout, I compared predictions made with and without dropout active at test time (Figure 4b). In `model.eval()` mode (standard dropout), predictions remain overconfident across all rotations. With MC Dropout, however, uncertainty is reflected in the spread of predicted classes and lower softmax probabilities — especially for rotations above $75°$.

Summary of Findings:

- **Uncertainty grows in out-of-distribution regions:** As shown in regression extrapolation and rotated digits.
- **Standard dropout is overconfident:** Predictive entropy remains low even on distorted inputs.
- **Tanh improves calibration:** Produces smoother transitions in predictive variance.
- **MC Dropout is sample-efficient:** Even $T = 10$ passes yield reasonable uncertainty estimates, although with more noise.

5 Conclusion

This work reproduced key experiments from Gal & Ghahramani (2016), confirming that MC Dropout enables uncertainty estimation in deep networks by approximating Bayesian inference with minimal changes to standard training.

In regression, MC Dropout captured uncertainty effectively, with variance increasing in out-of-distribution regions. Tanh activations provided smoother uncertainty estimates than ReLU. In classification, rotated MNIST experiments showed growing entropy with input distortion, highlighting the model's awareness of unfamiliar data.

A comparison with standard dropout revealed that `model.eval()` leads to overconfident predictions in unfamiliar regions which is an important limitation in safety-critical settings.

MC Dropout is practical: even $T = 10$ samples yielded usable uncertainty estimates. However, it's performance depends on factors like activation choice and sample count. It also only models epistemic uncertainty.

Overall, MC Dropout offers a simple, scalable, and theoretically grounded method for incorporating uncertainty into deep learning. Future work could extend it to modern architectures, combine it with aleatoric modeling, or apply it in active learning and decision-making systems.

References

[1] Gal, Y. & Ghahramani, Z. (2016). Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *ICML*.

[2] Paszke, A., et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *NeurIPS*.

[3] Harris, C.R., et al. (2020). Array programming with NumPy. *Nature*.

[4] Hunter, J.D. (2007). Matplotlib: A 2D Graphics Environment. *CiSE*.

[5] McKinney, W. (2010). Data Structures for Statistical Computing in Python. *Proc. Python Sci. Conf.*