UQ in Deep Learning using Bootstrap

Ananya Roy* Connor McNeill* Carter Hall*

April 22, 2025

Abstract

While deep neural networks have impressive predictive accuracy, they can often be overconfident. As such, uncertainty quantification is needed in order to assess the reliability of these predictions. We explore many approaches to performing this, including Bayesian approaches such as Monte Carlo (MC) Dropout, but we will focus on utilizing the Bootstrap technique for UQ in these deep learning models. A simulation study is conducted to apply these methods and assess performance. Overall, dropout rate greatly can impact results between the different methods. When there is more noise in the data, MC Dropout tends to perform better than bootstrap ensembles. Code is available at the Github repository linked here.

1 Background

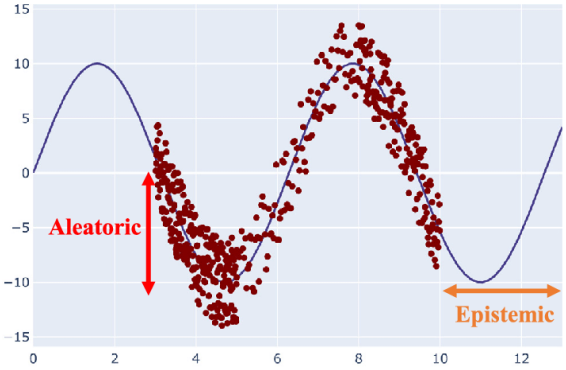
Deep learning has emerged as one of the most transformative machine learning techniques of the past decade. It has enable major advancements in domains such as natural language processing, computer vision, scientific computing, and healthcare. Much of deep learning relies on using neural networks with multiple (three or more) computational layers to train the models. We refer to these as deep neural networks (DNNs). While modern deep neural networks exhibit impressive predictive accuracy, they are often prone to producing overconfident predictions, even on out-of-distribution inputs. This lack of calibrated uncertainty can severely hinder their application in high-stakes or safety-critical domains. Consequently, **uncertainty quantification** (UQ) has become a vital component in designing trustworthy and robust deep learning models.

UQ in deep learning seeks to assess how reliable a prediction is by estimating a probability distribution or confidence interval over the model outputs. There are two main types of uncertainties that UQ methods aim to capture:

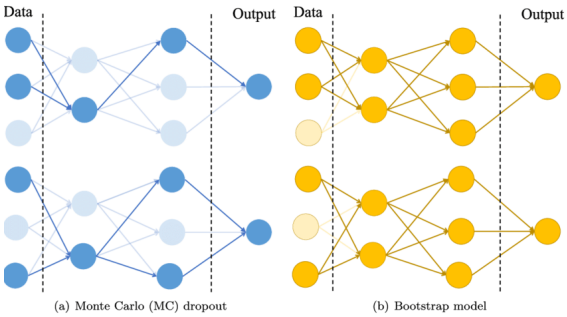
- **Aleatoric Uncertainty:** Commonly referred to as “statistical uncertainty”, it arises from inherent noise or stochasticity in the data generating process. It is *irreducible*, meaning that even with more data we cannot reduce the uncertainty. Examples include measurement noise and ambiguous class labels.
- **Epistemic Uncertainty:** Commonly referred to as “systematic uncertainty", it arises from uncertainty caused by a lack of knowledge about the best model. It is *reducible*, meaning that with more data we can reduce the uncertainty. Examples of what can cause this type of uncertainty include limited training data, insufficient model capacity, and out-of-distribution inputs.

Figure 1a shows these two types applied to predictive uncertainty. We see that Predictive Uncertainty can be broken into the two components, i.e. $PU = AU + EU$.

*Department of Statistics, North Carolina State University



(a) This figure from Abdar et al., 2021 shows graphically what the two types uncertainty are when applied to predictive uncertainty.



(b) This figure, also from Abdar et al., 2021, depicts the differences in neural network architectures between the methods of MC Dropout (left) and Bootstrap (right).

Figure 1: Discussion of Model/Data Uncertainty and MC Dropout

In this project, we focus on the epistemic (model) uncertainty via the **Bootstrap** method - a frequentist technique that approximates the sampling distribution of a statistic by resampling from the data. There are three types of bootstrapping that we will be utilizing as methods for uncertainty quantification:

- **Traditional (Nonparametric):** In nonparametric bootstrap, with resample from the data with replacement. Additionally, the size of each bootstrap sample is the same exact sample size as the original data. With a large number of bootstrap samples, we can use this to obtain sampling distribution of a certain statistic.
- **Bayesian:** In the Bayesian bootstrap, instead of obtaining a sampling distribution of a statistic, we simulate the posterior distribution of a parameter. This is the case because under the Bayesian framework, we assume that parameters are random variables and thus have a distribution.
- **Parametric:** For the parametric bootstrap, we make the assumption that the data follows a specific distribution and we use that to generate bootstrap samples. This can be computationally efficient but it requires assumptions about the parameters.

Traditional bootstrapping methods involve retraining the model on several bootstrapped datasets (i.e., samples drawn with replacement from the training set) and evaluating the ensemble of model predictions to estimate the variance. However, this naive approach is computationally intensive and often infeasible for large-scale DNNs.

To address these limitations, recent literature has proposed *efficient bootstrap-inspired methods* that approximate the benefits of bootstrapping without its computational burden. These include:

- Convexified Bootstrap for CNNs: Leveraging convex relaxation techniques to generate bootstrap ensembles efficiently. See Zhang et al., 2017 for a relevant work.
- Prediction Interval Optimized Bootstrap: A technique that trains NNs to directly construct tight prediction intervals. See Khosravi et al., 2015 for a relevant work.
- Calibrated Bootstrap Ensembles: Methods that correct underestimation of uncertainty using a post-hoc calibration step. See Palmer et al., 2022 for a relevant work.

In this report, we will first review some traditional UQ methods that have been utilized for deep learning models. We will then discuss how the Bootstrap technique is utilized for uncertainty quantification. The performance of these methods will be compared with a traditional method as a benchmark as part of a simulation study to explore the performance of these methods. Finally, we will discuss open problems and future work in this area.

2 Methods

2.1 Overview

First, it's important to explore the methodologies described in the literature. We reviewed multiple papers spanning the breadth of uncertainty quantification in deep learning, beginning with comprehensive and highly detailed survey papers that provide the theoretical backbone for this project. Abdar et al. (2021) provided a review of methods from a neural-network architecture perspective, while He et al. (2025) did the same but from a perspective of the type of uncertainty being quantified.

The methods can be organized into three main groups:

1. **Bayesian methods:** The goal of these methods is to implement the Bayesian framework to get a posterior distribution that can be utilized for uncertainty quantification.
2. **Frequentist methods:** The goal of these methods is to focus on a data distribution within a target population without incorporating prior beliefs or subjective probabilities.
3. **Ensemble-based methods:** The goal of these methods is to combine multiple neural networks to form a distribution, where the variability of the distribution is what quantifies uncertainty.

We will first explore the methodologies within these three groups as an overview. Then we will further examine the most popular "traditional" method along with the Bootstrap-based methods.

2.1.1 Bayesian Methods

Exact inference is intractable for deep networks, so approximations can be made via Variational Inference (VI) or Markov Chain Monte Carlo (MCMC). Variational Inference maximizes the evidence lower bound (ELBO) of the log marginal likelihood (He et al., 2025). Then the posterior distribution is approximated with a parametric distribution. However for deep learning there are some downsides with using Variational Inference: it can be computationally expensive due to the number of parameters in a deep learning model, along with it also relying on a choice of the parametric distribution that makes it intrinsically biased.

Within Bayesian methodology, MCMC is usually the standard choice for computing the posterior distribution. This method utilizes Monte Carlo sampling in order to generate samples which form our approximated posterior distribution. However the main challenge with using it is that it can be extremely expensive when it comes to computation. As such, alternative approaches are more common when it comes to UQ methods for deep learning.

The most popular Bayesian methodology utilized is Monte Carlo (MC) Dropout, which we will discuss in detail later on. It is often chosen due to its computational efficiency and ease of implementation. This method uses a single model and transforms it at inference time by randomly deactivating nodes within the neural network (Faghani et al., 2024). Figure 1b shows an example of what this dropout looks like.

Additional approaches include Bayesian Active Learning, which combines an active learning framework with a Bayesian neural network, and Bayes by Backprop, which quantifies the uncertainty of the weights within the neural network via VI (Abdar et al., 2021).

2.1.2 Frequentist Methods

Frequentist approaches treat the parameters as fixed and not as random variables like in the Bayesian framework. As such, it follows that a lot of the “standard" frequentist UQ approaches like confidence and prediction intervals capture only the aleatoric (data) uncertainty and not the epistemic uncertainty. Some modifications can be done to quantify the uncertainty in the model. One approach done by Khosravi et al. (2015) was to utilize the Bootstrap to obtain prediction intervals. This works as the Bootstrap can be used to estimate the model misspecification variance which can help quantify epistemic uncertainty.

Another frequentist approach is that is utilized conformal prediction. Conformal prediction is considered as distribution-free UQ, meaning that it is model agnostic, and provides formal results for marginal coverage (Gamble et al., 2024). This is quite helpful for deep learning models because of the traditional assumptions often being violated due to non-linearity and the complexity of the models. However, one downside to using this method is that exchangeability is required on the sample distribution, meaning that the data’s joint distribution remains unchanged under reordering. This is what ensures valid prediction sets with guaranteed coverage.

2.1.3 Ensemble-based Methods

Ensemble models are formed by aggregating multiple models to form a distribution that can be used to quantify the overall uncertainty in the model (He et al., 2025). Several approaches have been utilized to construct ensembles. One approach utilizes the Bootstrap to construct ensembles. This is our primary method that we will focus on throughout the paper. Bootstrapping is easy to implement and utilizing the ensemble-approach greatly improves the UQ compared to not creating ensembles. We’ll discuss this method in more detail later on.

Another way to form ensembles is by varying the neural network architecture. This includes the number of layers, how many neurons the hidden layers have, and the types of activation functions utilized (Mallick et al., 2022). This all combines together to create an ensemble of neural networks, accounting for the uncertainty from model misspecification. Other approaches include random shuffling of datasets, differing parameter initializations, and also the hyper-ensemble approach which constructs ensembles by varying hyper-parameters.

Often, two evaluation measures are applied in order to ensure quality in predictive uncertainty quantification. The first is calibration, which measures the discrepancy between frequencies and forecasts. Also there is the notion concerning domain shifts that estimates whether the neural network truly knows what it knows (Abdar et al., 2021).

There are a couple of downsides with using ensemble-based methods. The first is that it is computationally expensive due to requirement of needing multiple neural networks. Additionally, it requires model diversity in order to actually make sure that our uncertainty quantification is accurate (He et al., 2025).

2.2 Monte Carlo (MC) Dropout

As previously mentioned above, Monte Carlo (MC) Dropout is one of the most popular methods for uncertainty quantification in deep learning. Since it is an approximate Bayesian method, it is much more computationally efficient compared to utilizing Markov Chain Monte Carlo (MCMC). Additionally there are some advantages compared to other UQ methods for deep learning:

- It works with the existing deep neural network architecture. Unlike some of the ensemble methods we looked that, this means that MC Dropout is quite simple to implement.
- We mitigate the problem of representing uncertainty by sacrificing model accuracy since it only impacts the inference.

However, when compared with other methods, MC Dropout tends to be less calibrated when applied to several uncertainty benchmark datasets (Guo et al., 2017). Additionally, since it falls under the umbrella of *approximate* Bayesian computing (ABC), it means that for complex posterior distributions it may not do the best with approximation. Nonetheless, for most common situations, the advantages mentioned make MC Dropout a good choice as a method.

First, it is important to understand what dropout is. Dropout is a technique utilized to address the overfitting of neural network. As developed by Srivastava et al. (2014), the key goal is to randomly drop nodes (along with their connections) from the neural network while training. In simplest cases, the dropout probability can be set to $p = 0.5$, however, validation sets can also be utilized to determine the optimal rate. For input layers, the optimal probability of retention is closer to 1 (meaning that p is closer to 0) than to 0.5. Figure 1b shows an example of dropout being applied to a deep neural network. Not only does this prevent overfitting, it becomes much more computationally efficient than ensemble methods. Yet it still provides a way of approximately combining exponentially many different neural networks. This also means when applying Monte Carlo sampling to this as this means it is computationally less expensive than sampling from the entire neural network.

Gal and Ghahramani (2016) shows that a neural network with dropout applied before every weight layer is mathematically equivalent to a deep Gaussian Process (GP). In the interest of being brief, we will omit most of the mathematical theory that shows that MC Dropout works. Gal and Ghahramani (2016) goes into detail with that. In short, we utilize the neural network with dropout to approximate a deep GP, and then we use Monte Carlo integration to approximate the posterior distribution.

2.3 Bootstrap Ensembles

As discussed earlier, utilizing bootstrap ensembles is one approach that involves training multiple models on resampled versions of the original dataset. This resampling is done by bootstrapping, which can be done either with a parametric, nonparametric, or Bayesian approach. The variability across the models' predictions is what is used to quantify uncertainty.

Bootstrap ensembles work utilizing the following mechanism:

1. We first utilize bootstrapping to create multiple datasets.
2. Next, we train independent deep neural networks on each resampled dataset.
3. We then aggregate the predictions into our ensemble.
4. Lastly we use this ensemble and compute our uncertainty estimation.

In our case, we will compute the bootstrap standard deviation. This measure happens to be a great way to quantify uncertainty within our model (epistemic uncertainty).

There are several advantages to utilizing this method. First, like MC Dropout, it is simple to implement due to there being no architectural changes need to the neural networks. It also is easy to interpret as we get a common measure of uncertainty (in our case the standard deviation). Additionally, it is parallelizable, meaning that we can train the models independently.

However, there are a few disadvantages to using this method as well. First, training multiple models at once can be computationally expensive. Additionally, bootstrap is much better suited for epistemic (model) uncertainty as it underestimates the aleatoric uncertainty. Finally, there can be high similarity in the bootstrapped samples which limits the model diversity needed in an ensemble-based method. Overall, it still is a method that is not only easy to implement but also is quite interpretable, hence making it extremely useful.

2.3.1 Calibration

In Palmer et al., 2022, the authors point out that "the direct bootstrap ensemble standard deviation is not an accurate estimate of uncertainty but... can be simply calibrated to dramatically improve its accuracy." What they propose is an analog to a method that assumes the squared estimates, the "prediction variances $\widehat{\sigma}^2(x)$," are linearly related to distance-based uncertainty estimates $U(x)$ (e.g., how far a test datum is from training data or decision boundaries, depending on the regime of problem) – thus, $\widehat{\sigma}^2(x) = aU(x) + b$ for some $a, b \in \mathbb{R}$. Under their assumption that any validation prediction $\widehat{y}_{\text{val}} \sim N(y_{\text{val}}, \widehat{\sigma}^2(x_{\text{val}}))$, where y_{val} denotes the true response value for observation x_{val}, one simply optimizes for a, b by taking the gradient of the product of likelihoods of [assumed independent] observations from the validation set. What Palmer et al., 2022 shows through visualizations is that the values of a, b obtained from solving this optimization problem, when used to scale standard deviation estimates obtained from a bootstrap procedure (denoted $\widehat{\sigma}_{\text{uc}}$) through $\widehat{\sigma}_{\text{cal}} = a\widehat{\sigma}_{\text{uc}} + b$, the calibrated estimates will exhibit more desired properties (e.g., coverage) and not over/under-estimate the true standard deviation.

In our simulation studies, calibration will be a component, but not the focus, of the analysis. Because the project is primarily focused on uncertainty quantification through the bootstrap, we will be comparing aforementioned bootstrap procedures. Calibration will be used as reassurance that the ensemble procedures do not wildly differ in their UQ; that is, to say, that the confidence intervals surrounding mean predictions for each bootstrap method do not markedly differ in the areas they cover. This behvaior of course is *expected*, but any deviation from this expectation is cause for further discussion.

3 Simulation Study

Because our project focuses on deep learning, it is only natural for the model on which our bootstrap procedures will be evaluated to be a deep-learning model. With this in mind, we propose using a rather simple architecture, detailed below:

- **Input Layer**: This layer will have dimension of the number of covariates. For example, in our 3.1 regression example, we supply polynomial powers of uniformly simulated data up to fifth-order. As such, the input layer will have five nodes in this example.
- **First Deep Layer**: A layer of 50 neurons with a hyperbolic-tangent activation function.

- **Second Deep Layer**: A layer of 25 neurons with a hyperbolic-tangent activation function.
- **Dropout Layer**: Parameterized by a dropout rate in $(0, 1)$, this layer will zero out connections between the previous deep layer and the succeeding output layer.
- **Output Layer**: A layer with a single node, as all of our simulation studies have univariate, continuous response.

Our simulation study for this regression problem will proceed as follows:

1. For $M = 10$ Monte Carlo replicates, generate a dataset and perform MC Dropout and five (5) Bootstrap Ensemble procedures. The choice of $M = 10$ is as much due to 'proof-by-concept' as it is to one of our group member's computer breaking over the weekend, impeding progress greatly!
2. At each replicate, take the average of the mean predictions on the *test set* across the number of MC Dropout forward passes or the number of ensembles in a bootstrap procedure. Do the same for the calibrated standard deviations.
3. Additionally, take the standard deviation *of* the aggregated standard deviations in the step above.

Why do this? Performing this procedure will generate for each data point in the test dataset the following items: an average of mean predictions, an average standard deviation for each prediction, and the standard deviation *of* the standard deviation at each prediction... for each of the four methods.

What comes from this procedure? With these results computed, we will be able to effectively visualize uncertainty quantification estimates. Because the calibrated standard deviations are inherently functions of the noised responses in the validation set, they are random. As such, taking a standard deviation of these estimates will allow us to visualize the variability in the calibration step when applied to the same bootstrap procedure on the same problem. We will compare the results based on two error criterion – Mean Squared Error and Mean Absolute Error. Both are sensible error criterion in a regression context.

3.1 Simple Case: Regression

To study the potential of various bootstrap methods for uncertainty quantification in a regression problem, we began efforts to perform computer simulations with a simple, easily reproducible/scalable task of estimating the function $y = \sin(x)$ with $x \in \mathbb{R}$ and $y \in [-1, 1]$. Of course, such a problem is far too simplistic – we were able to, separate from our simulation study and project, naturally perform multiple linear regression on powers of x to estimate the function. This is, of course, due to the well-known result that, through Taylor expansion, $\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots$.

To introduce data (aleatoric) uncertainty, we added Gaussian noise to perturb the responses around their true function values. Because we additionally wanted to compare the three bootstrap approaches – Standard, Bayesian, and Parametric – to a 'baseline' method, MC Dropout was chosen to be the basis of our comparison. The reason for this choice comes from He et al., 2025 in which MC Dropout is described as a method very easy to implement and, as such, much to our liking.

Our simulation study for this simpler regression problem comes in two flavors:

1. We increase the dropout rate between the final (second) deep layer and the output layer across values 0.1, 0.3, 0.5, ..., 0.9.

2. We fix the dropout rate at 0.1 and vary the amount of noise added to the data, taking values $\sigma \in \{0.25, 0.5, 1, 2\}$. This is inspired by the work of Palmer et al., 2022 who did a very similar study on a more complex function – while their function is more complex, our decision to stick with a simpler problem in this report is so that we can easily visualize the methods' performance of regressing the noisy response y on data x. If the data $\boldsymbol{x}$ were vector-valued, a cross-section of the high-dimensional surface visualizing y as a function of $\boldsymbol{x}$ would be naturally difficult to elegantly visualize in a 2D graph. (This was experienced when replicating their results as a test, but dialed back to the scale presented for our report.)

3.1.1 Varying Dropout Rate

Due to the sheer volume of results and the desire to stick within the page limit, we will present results and note where there exist major similarities across the bootstrap procedures. Of particular interest in this section is how changing the dropout rate in the deep learning architecture affected method performance under fixed noise.

Through Table 1, we see great similarity in the error metrics when the dropout rate is relatively low, e.g. 0.1. This is arguably expected because the architecture is able to effectively characterize the true sinusoidal relationship in the data when dropout does nut suddenly alter the predictions in a drastic manner. We see this corroborated in Figure 2 where the green 'squares' on the upper curves are actually standard error bars (in orange) around every twentieth observation in the test set. As the dropout rate increases to 0.5, we interestingly see those bars around the upper 95% confidence limit appear (we chose to omit showing for the lower limit for brevity) much clearer. This suggests the dropout has introduced greater variability in the model predictions. Of course, if we incorporated a larger number of ensembles into each visualization, this would perhaps approach some limiting shape. However, the drastic change from Figure 2 to Figure 3 is interesting.

Another interesting result occurs when the dropout rate is set at 0.9, shutting off a majority of the connections between the final hidden layer and the output layer. Figure 4 shows that even the MC Dropout method, a method praised for simplicity and accuracy, has its behavior massively influenced. Interestingly, the Bayesian and Standard Bootstrap procedures were relatively robust to the increased dropout rate. The authors express confidence in the code used to perform the simulations – the only value changed throughout the code for this subsection was the dropout rate. Table 2 supports this behavior – we see the error metrics for parametric bootstrap become *significantly* worse!

	MSE	MAE	Avg Uncertainty
Bayesian	0.062640	0.202347	0.249944
Standard	0.061840	0.200689	0.251773
Parametric	0.065412	0.205807	0.256946

Table 1: Comparison of Mean Squared Error (MSE) and Mean Absolute Error (MAE) for Bootstrap Approaches when Dropout Rate = 0.1

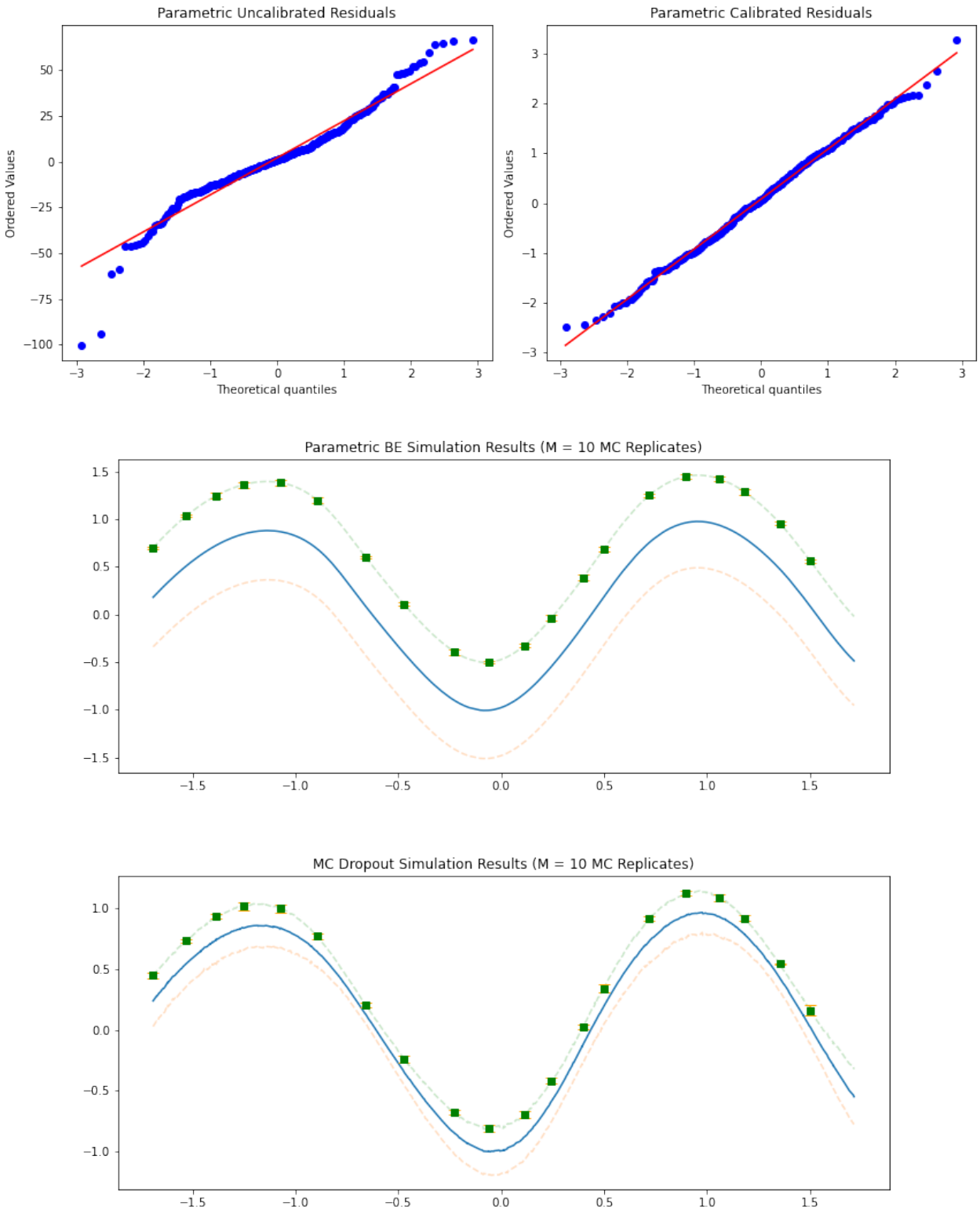
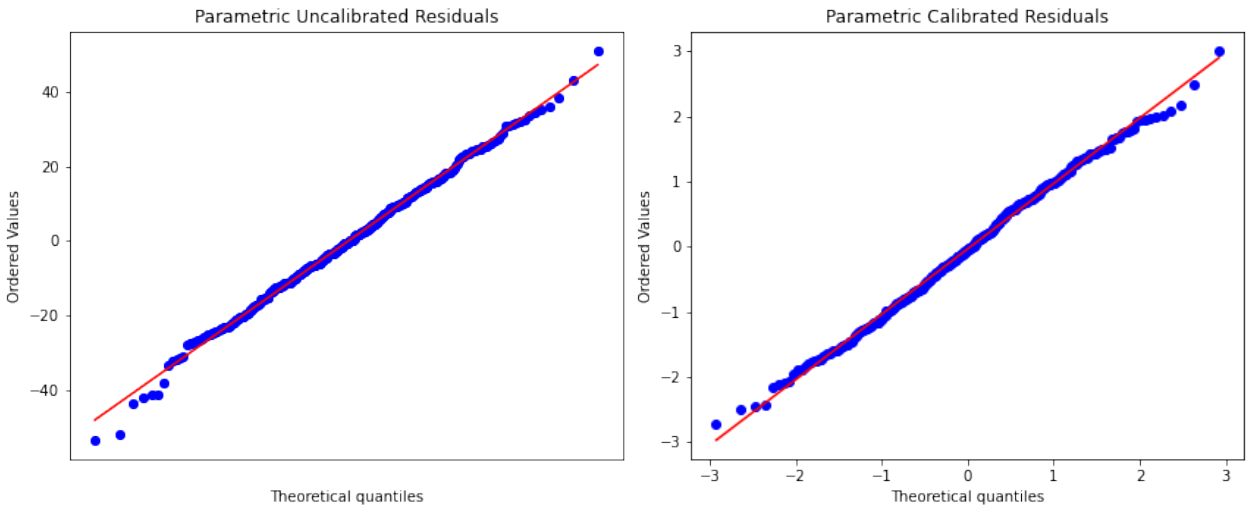


Figure 2: (Top) Illustration of Calibration of Residuals for Parametric Bootstrap. (Bottom) Comparison of Parametric Bootstrap and Monte Carlo Dropout Results for $M = 10$ Replicates at a Dropout Rate of 0.1. Standard and Bayesian Bootstrap results were ***very*** similar to Parametric. Orange and green dotted curves represent 95% CI.



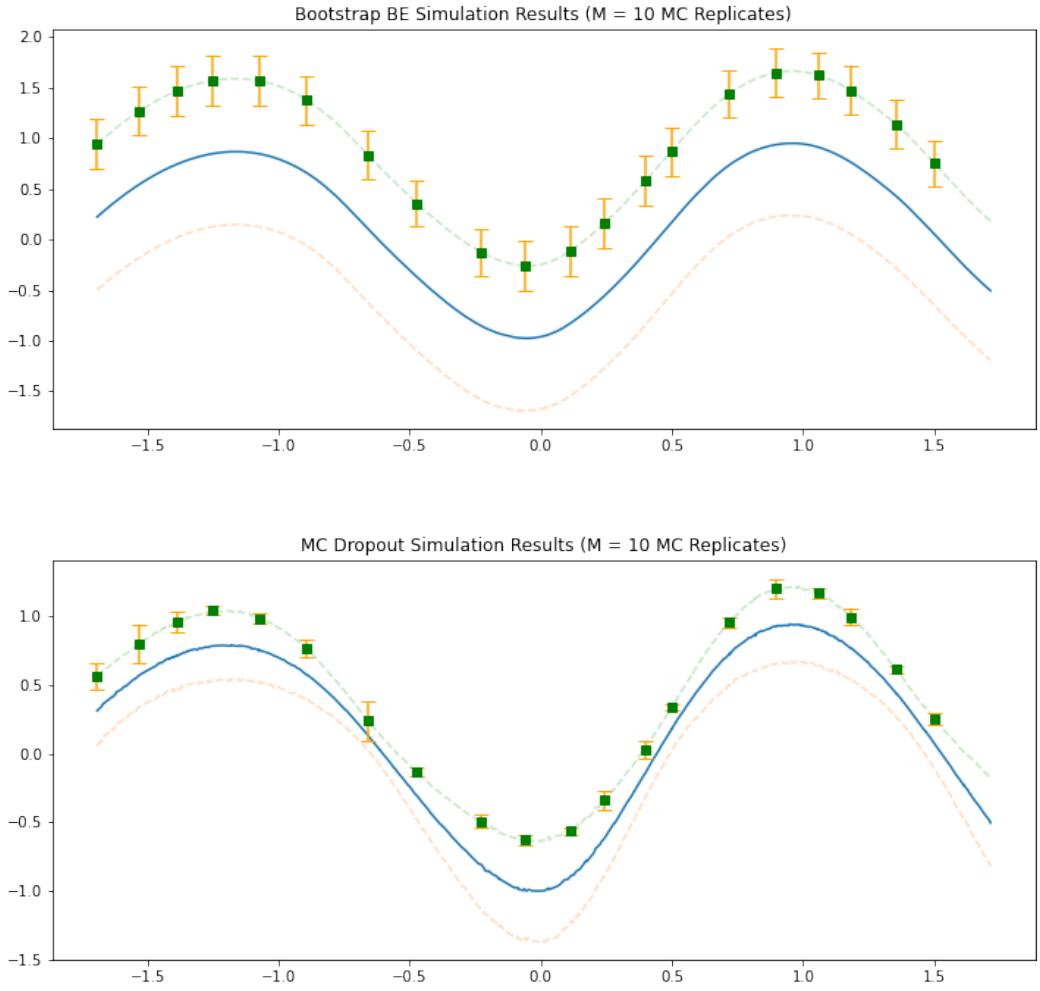


Figure 3: (Top) Illustration of Calibration of Residuals for Parametric Bootstrap. (Bottom) Comparison of Parametric Bootstrap and Monte Carlo Dropout Results for $M = 10$ Replicates at a Dropout Rate of 0.5. Standard and Bayesian Bootstrap results were ***very*** similar to Parametric. Orange and green dotted curves represent 95% CI.

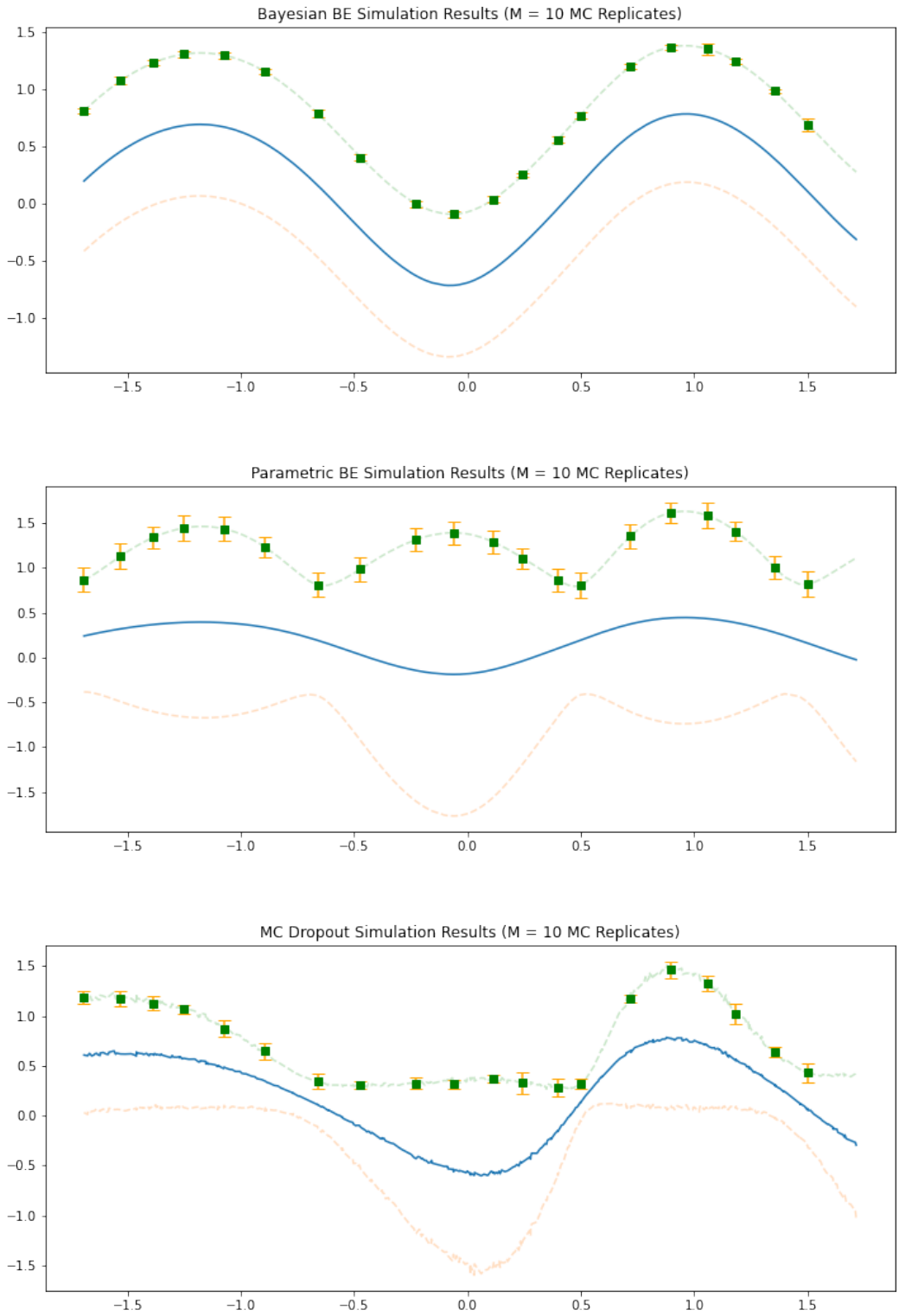


Figure 4: Comparison of Bayesian, Parametric Bootstrap and Monte Carlo Dropout Results for $M = 10$ Replicates at a Dropout Rate of 0.9. Standard Bootstrap results were ***very*** similar to Bayesian, but Parametric performed much worse. Orange and green dotted curves represent 95% CI.

	MSE	MAE	Avg Uncertainty
Bayesian	0.094985	0.246860	0.311934
Standard	0.100423	0.253728	0.325025
Parametric	0.275670	0.442868	0.511634

Table 2: Comparison of Mean Squared Error (MSE) and Mean Absolute Error (MAE) for Bootstrap Approaches when Dropout Rate = 0.9

3.1.2 Varying Noise Added to Response

As a separate angle to our analysis, we fix the dropout rate at 0.1 and instead increase the variance of the noise added to the generated data. Figure 5 illustrates the effects; noticeably, the confidence intervals have greatly increased in width due to the noise. Of course, with responses of $y = \sin(x) \in [-1, 1]$, adding such a noisy component to the data is expected to dominate the analysis. This is the case for the bootstrap ensemble methods, as they perform far worse relative to the MC dropout methods in this regime.

4 Discussion

4.1 Conclusion

This project focused on uncertainty quantification for deep learning models, particularly via the bootstrap. We extend beyond the classical bootstrap method to introduce and investigate other techniques such as the Bayesian and Parametric bootstrap. Our regression example showcases how, relative to the standard of MC Dropout, bootstrap ensemble methods can outperform in cases of high dropout rate but greatly struggle when noise obfuscates the relationship between data and response.

4.2 Future Work/Open Problems

A notable area of future work is applying our simulations to a real-world dataset, e.g., the California Housing Prices dataset. The only impediment to this being completed prior to project deadline was the aforementioned issue of a group member's computer being physically broken for days, delaying simulation progress. Another area of future work would be to leverage the application of our simulation and bootstrapping procedures to more difficult scenarios in order to study the effects of calibration more intensely. We see that for our simple regression example the calibrated standard deviations vary little relative to their mean, though this may not continue in a more complex problem (e.g., learning a more difficult function). Palmer et al., 2022 suggest performing a convergence analysis where the number of ensembles for a bootstrap procedures is increased until the optimized a, b values converge. This was *attempted* in earlier stages of the project but, due to technological issues and computational expense, was not focused on further.

Other open problems for utilizing bootstrap for UQ include scaling to modern deep learning models that are high-dimensional, working on handling aleatoric (data) uncertainty in addition to epistemic uncertainty, and integration with modern architectures such as large-language models (LLMs). Since deep learning is such a popular area right now with the rise of AI, uncertainty quantification is increasingly important, hence the need for methods that are accurate and interpretable.

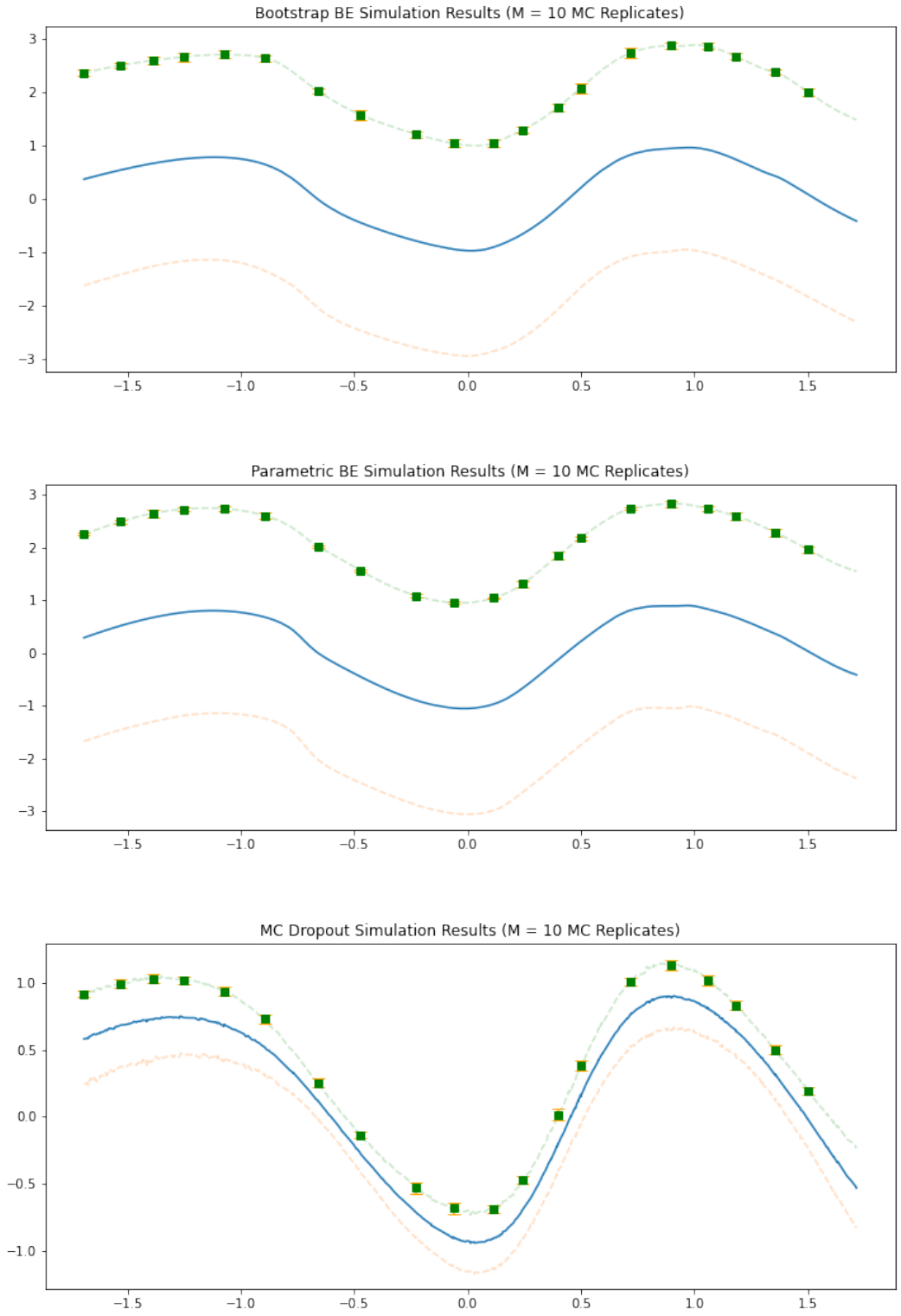


Figure 5: (Top) Illustration of Calibration of Residuals for Parametric Bootstrap. (Bottom) Comparison of Parametric Bootstrap and Monte Carlo Dropout Results for $M = 10$ Replicates when noise variance increased to 1 from 0.25. Standard and Bayesian Bootstrap results were ***very*** similar to Parametric. Orange and green dotted curves represent 95% CI.

References

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., et al. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, *76*, 243–297. https://doi.org/10.1016/j.inffus.2021.05.008

Faghani, S., Gamble, C., & Erickson, B. J. (2024). Uncover This Tech Term: Uncertainty Quantification for Deep Learning. *Korean J Radiol*, *25*(4), 395–398. https://doi.org/10.3348/kjr.2024.0108

Gal, Y., & Ghahramani, Z. (2016, 20–22 Jun). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In M. F. Balcan & K. Q. Weinberger (Eds.), *Proceedings of the 33rd international conference on machine learning* (pp. 1050–1059, Vol. 48). PMLR. https://proceedings.mlr.press/v48/gal16.html

Gamble, C., Faghani, S., & Erickson, B. J. (2024). Toward clinically trustworthy deep learning: Applying conformal prediction to intracranial hemorrhage detection. https://doi.org/10.48550/arXiv.2401.08058

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017, June). On calibration of modern neural networks. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international conference on machine learning* (pp. 1321–1330, Vol. 70). PMLR. https://proceedings.mlr.press/v70/guo17a.html

He, W., Jiang, Z., Xiao, T., Xu, Z., & Li, Y. (2025). A Survey on Uncertainty Quantification Methods for Deep Learning. (arXiv:2302.13425). https://doi.org/10.48550/arXiv.2302.13425

Khosravi, A., Nahavandi, S., Srinivasan, D., & Khosravi, R. (2015). Constructing Optimal Prediction Intervals by Using Neural Networks and Bootstrap Method. *IEEE Transactions on Neural Networks and Learning Systems*, *26*(8), 1810–1815. https://doi.org/10.1109/TNNLS.2014.2354418

Mallick, T., Balaprakash, P., & Macfarlane, J. (2022). Deep-ensemble-based uncertainty quantification in spatiotemporal graph neural networks for traffic forecasting. https://doi.org/10.48550/arXiv.2204.01618

Palmer, G., Du, S., Politowicz, A., Emory, J. P., Yang, X., Gautam, A., Gupta, G., Li, Z., Jacobs, R., & Morgan, D. (2022). Calibration after bootstrap for accurate uncertainty quantification in regression models. *npj Computational Materials*, *8*(1), 1–9. https://doi.org/10.1038/s41524-022-00794-8

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, *15*(1), 1929–1958.

Zhang, Y., Liang, P., & Wainwright, M. J. (2017). Convexified Convolutional Neural Networks. *Proceedings of the 34th International Conference on Machine Learning*, 4044–4053. Retrieved February 11, 2025, from http://proceedings.mlr.press/v70/zhang17f/zhang17f.pdf