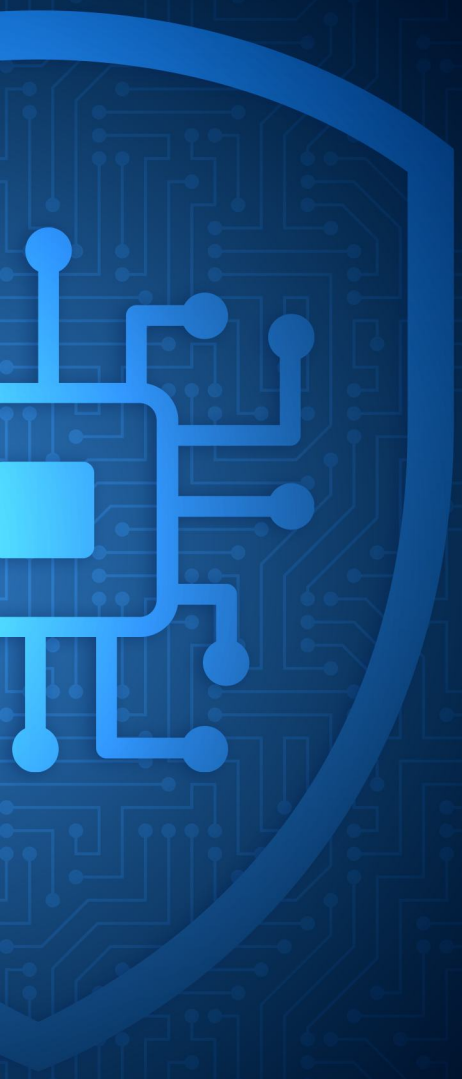
Issue Brief

Key Concepts in AI Safety

Reliable Uncertainty Quantification in Machine Learning



Authors
Tim G. J. Rudner
Helen Toner

CSET CENTER for SECURITY and EMERGING TECHNOLOGY

June 2024

This paper is the fifth installment in a series on "AI safety," an area of machine learning research that aims to identify causes of unintended behavior in machine learning systems and develop tools to ensure these systems work safely and reliably. Other papers in the series describe three categories of AI safety issues—problems of robustness, assurance, and specification. This paper introduces the idea of uncertainty quantification, i.e., training machine learning systems that "know what they don't know."

Introduction

The last decade of progress in machine learning research has given rise to systems that are surprisingly capable but also notoriously unreliable. The chatbot ChatGPT, developed by OpenAI, provides a good illustration of this tension. Users interacting with the system after its release in November 2022 quickly found that while it could adeptly find bugs in programming code and author *Seinfeld* scenes, it could also be confounded by simple tasks. For example, one dialogue showed the bot claiming that the fastest marine mammal was the peregrine falcon, then changing its mind to the sailfish, then back to the falcon—despite the obvious fact that neither of these choices is a mammal. This kind of uneven performance is characteristic of deep learning systems—the type of AI systems that have seen most progress in recent years—and presents a significant challenge to their deployment in real-world contexts.

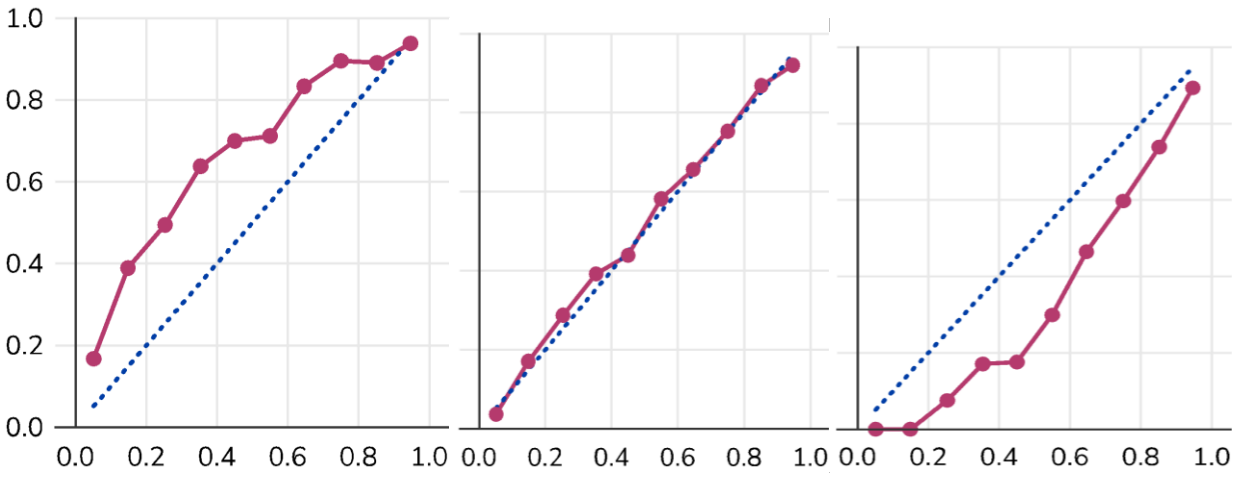
An intuitive way to handle this problem is to build machine learning systems that "know what they don't know"—that is, systems that can recognize and account for situations where they are more likely to make mistakes. For instance, a chatbot could display a confidence score next to its answers, or an autonomous vehicle could sound an alarm when it finds itself in a scenario it cannot handle. That way, the system could be useful in situations where it performs well, and harmless in situations where it does not. This could be especially useful for AI systems that are used in a wide range of settings, such as large language models (the technology that powers chatbots like ChatGPT), since these systems are very likely to encounter scenarios that diverge from what they were trained and tested for.

Unfortunately, designing machine learning systems that can recognize their limits is more challenging than it may appear at first glance. In fact, enabling machine learning systems to "know what they don't know"—known in technical circles as "uncertainty quantification"—is an open and widely studied research problem within machine learning. This paper gives an introduction to how uncertainty quantification works, why it is difficult, and what the prospects are for the future.

The Challenge of Reliably Quantifying Uncertainty

In principle, the kind of system we would like to build sounds simple: a machine learning model that generally makes correct predictions, but that can indicate when its predictions are more likely to be incorrect. Ideally, such a model would indicate high levels of uncertainty neither too often nor too seldom. A system that constantly expresses under-confidence in situations that it could actually handle well is not very useful, but if the system sometimes does not indicate uncertainty when in fact it is about to fail, then this defeats the purpose of trying to quantify uncertainty in the first place. Experts use the idea of "calibration" to describe the desired behavior here: the level of uncertainty that a machine learning model assigns to a given prediction—its "predictive uncertainty"—should be calibrated to the probability that the prediction is in fact incorrect.

Figure 1: Calibration Curves Depicting Under-Confidence, Near-Perfect Calibration, and Over-Confidence



The figures show under-confident (left), well-calibrated (center), and over-confident (right) calibration curves. Ideally, the confidence expressed by the model (on the x-axis) should correspond to the chance that the prediction is correct (on the y-axis). A model is under-confident if its predictions are more often correct than its confidence levels would imply (per the chart on the left), while the inverse is true for an over-confident model (on the right).

Source: CSET.

For example, imagine a medical machine learning classification system that uses a scan of a patient's eye to predict whether the patient has a retinal disease.[1] If the system is calibrated, then its predictions—typically expressed as percentages—should correspond to the true proportion of diseased retinas. That is, it should be the case that

of the retina images predicted to be exhibiting signs of disease with a 50% chance, half are in fact diseased, or that eight out of ten retina images predicted to have an 80% probability of exhibiting signs of disease in fact do, and so on. The closer the assigned probabilities are to the real proportion in the evaluation data, the better calibrated the system is. A well-calibrated system is useful because it allows users to account for how likely the prediction is to be correct. For example, a doctor would likely make different decisions about further testing and treatment for a patient whose scan indicated a 0.1% chance of disease versus one whose scan indicated a 30% chance—even though neither scan would be classified as likely diseased.

Understanding Distribution Shift

Building a system that can express well-calibrated predictive uncertainty in the laboratory—while not straightforward—is achievable. The challenge lies in creating machine learning models that can reliably quantify uncertainty when subjected to the messiness of the real world in which they are deployed.

At the root of this challenge lies an idea called "distribution shift." This refers to the ways in which the types of data that a machine learning system encounters (the "data distribution") change from one setting to another. For instance, a self-driving car trained using data from San Francisco's roads is unlikely to encounter snow, so if the same car were deployed in Boston during the winter, it would encounter a different data distribution (one that includes snow on the roads), making it more likely to fail.

Distribution shift is easy to describe informally, but very difficult to detect, measure, or define precisely. This is because it is especially difficult to foresee and account for all the possible types of distribution shifts that a system might encounter in practice. When a particular shift can be anticipated—for instance, if the engineers that trained the self-driving car in San Francisco were planning a Boston deployment and considering weather differences—then it is relatively straightforward to manage. In most cases, however, it is impossible to know in advance what kinds of unexpected situations—what unknown unknowns—a system deployed in the messy real world may encounter.

The need to deal with distribution shifts makes quantifying uncertainty difficult, similarly to the broader problem of generalization in modern machine learning systems. While it is possible to evaluate a model's accuracy on a limited set of data points in the lab, there are no mathematical guarantees that ensure that a model will perform as well when deployed (i.e., that what the system learned will "generalize" beyond its training data). Likewise, for uncertainty quantification, there is no guarantee

that a seemingly well-calibrated model will remain calibrated on data points that are meaningfully different from the training data. But while there is a vast amount of empirical and theoretical literature on how well models generalize to unseen examples, there is relatively little work on models' ability to reliably identify situations where their uncertainty should be high, making "uncertainty generalization" one of the most important and yet relatively underexplored areas of machine learning research.

Accurately Characterizing Uncertainty

In the medical imaging example above, we described how machine learning models used for classification produce probabilities for each class (e.g., diseased versus not diseased), but such probabilities may not be sufficient for reliable uncertainty quantification. These probability scores indicate how strongly a model predicts that a given input corresponds to a given output. For instance, an image classifier for reading zip codes takes in an image of a handwritten digit, then assigns a score to each of the ten possible outputs (corresponding to the digit in the image being a "0," "1," "2," etc.). The output with the highest score indicates the digit that the classifier thinks is most likely to be in the image.

Unfortunately, these scores are generally not useful indicators of the model's uncertainty, for two reasons. First, they are the result of a training process that was optimizing for the model to produce accurate outputs, not calibrated probabilities;[2] thus, there is no particular reason to believe that a score of 99.9% reliably corresponds to a higher chance that the output is correct than a score of 95%. Second, systems designed this way have no way to express "none of the above"—say, if the zip code reader encountered a bug splattered across the page. The model is mathematically forced to assign probability scores to the available outputs, and to ensure that those scores sum to one.[3]

This naturally raises the question of why adding a "none of the above" option is not possible. The reason is simple: models learn from data and, due to the challenges of distribution shift described above, AI developers typically do not have data that represents the broad range of possibilities that could fit into a "none of the above" option. This makes it infeasible to train a model that can consistently recognize inputs as being meaningfully different.

To summarize, the core problem making uncertainty quantification difficult is that in many real-world settings, we cannot cleanly articulate and prepare for every type of situation a model may need to be able to handle. The aim is to find a way for the system to identify situations when it is likely to fail—but because it is impossible to

expose the system to every kind of scenario in which it might perform poorly, it is impossible to verify in advance that the system will appropriately estimate its chances of performing well under novel, untested conditions. In the next section, we discuss several approaches that try to navigate this difficulty.

Existing Approaches to Uncertainty Quantification

The key challenge of uncertainty quantification is to develop models that can accurately and reliably express how likely their predictions are to be correct. A wide range of approaches have been developed that aim to achieve this goal. Some approaches primarily treat uncertainty quantification as an engineering challenge that can be addressed with tailored algorithms and more training data. Others seek to use more mathematically grounded techniques that could, in theory, provide watertight guarantees that a model can quantify its own uncertainty well. Unfortunately, it is not currently possible to produce such mathematical guarantees without using unrealistic assumptions. Instead, the best we can do is develop models that quantify uncertainty well on carefully designed empirical tests.

Approaches to uncertainty quantification in modern machine learning fall into four different categories:

1. Deterministic Methods
2. Model Ensembling
3. Conformal Prediction
4. Bayesian Inference

Each of these approaches has distinct benefits and drawbacks, with some providing mathematical guarantees and others performing particularly well on empirical tests. We elaborate on each technique in the remainder of this section. Readers are welcome to skip to the next section if the somewhat more technical material below is not of interest.

Deterministic Methods

Deterministic methods work by explicitly encouraging the model to exhibit high uncertainty on certain input examples during training. For example, researchers might start by training a model on one dataset, then introduce a different dataset with the expectation that the model should express high uncertainty on examples from the dataset it was not trained on. Using this approach results in models that are very accurate on data similar to what they were trained on, and that indicate high uncertainty for other data.[4]

However, it is not clear how much we can rely on these research results in practice. Models trained this way are optimized to recognize that some types of input are outside the scope of what they can handle. But because the real world is complex and unpredictable, it is impossible for this training to cover all possible ways in which an input could be out of scope. For example, even if we trained the medical imaging classifier described above to have high predictive uncertainty on images that exhibit commonly known image corruptions, it may still fail at deployment if the model was trained on images obtained in one hospital with a certain type of equipment, and deployed in another hospital with a different type of equipment. As a result, this approach is prone to failure when the model is deployed, and there is no known way to guarantee that the predictive uncertainty estimates will in fact be reliable.

Model Ensembling

Model ensembling is a simple method that combines multiple trained models and averages their predictions. This approach often improves predictive accuracy compared to just using a single model. An ensemble's predictive uncertainty is expressed as the standard deviation of the different predictions, meaning that if all of the models in the ensemble make similar predictions, then uncertainty is low; if they make very different predictions, uncertainty is high. Ensemble methods are often successful at providing good predictive uncertainty estimates in practice, and are therefore a popular approach—though they can be expensive, given that multiple models must be trained. The underlying mechanism of using ensembling for uncertainty quantification is that different models in an ensemble will be likely to agree on input examples similar to the training data, but may disagree on input examples meaningfully different from the training data. As such, when the predictions of the ensemble components differ, this can be used as a stand-in for uncertainty.[5]

However, there is no way to verify that this mechanism works for any given ensemble and input example. In particular, it is possible that for some input examples, multiple models in the ensemble may all give the same incorrect answer, which would give a false impression of confidence, and it is impossible to ensure that a given ensemble will provide reliable, well-calibrated predictive uncertainty estimates across the board. For some use cases, the fact that ensembling typically provides fairly good uncertainty estimates may be sufficient to make it worth using. But in cases where the user needs to be able to trust that the system will reliably identify situations where it is likely to fail, ensembling should not be considered a reliable method.

Conformal Prediction

Conformal prediction, in contrast with deterministic methods and ensembling, is a statistically well-founded approach that provides mathematical reliability guarantees, but relies on a key assumption: that the data the model will encounter once deployed is generated by the same underlying data-generating process as the training data (i.e., that there is no distribution shift). Using this assumption, conformal prediction can provide mathematical guarantees of the probability that a given prediction range included the correct prediction. For instance, in a weather forecasting setting, conformal prediction could guarantee a 95% chance that the day's maximum temperature will fall within a certain range. (That is, it could provide a mathematical guarantee that 95 out of 100 similar predictions would fall within the range.)[6] A predicted range of, say, 82°F-88°F would imply more uncertainty than a range of 83°F-85°F.

Conformal prediction's major advantage is that it is possible to mathematically guarantee that its predictive uncertainty estimates are correct under certain assumptions. Its major disadvantage is that those assumptions—primarily that the model will encounter similar data while deployed to the data it was trained on—often do not hold. Worse, it is often impossible to detect when these assumptions are violated, meaning that the same kind of changes in inputs that may trip up deterministic methods are also likely to cause conformal prediction to fail. In fact, in all of the example application problems where machine learning models are prone to fail and for which we would like to find approaches to improving uncertainty quantification, standard assumptions of conformal prediction would be violated.

Bayesian Inference

Lastly, Bayesian uncertainty quantification uses Bayesian inference, which provides a mathematically principled framework for updating the probability of a hypothesis as more evidence or information becomes available.[7] Bayesian inference can be used to train a neural network that represents each parameter in the network as a random variable, rather than a single fixed value (as is typically the case). While this approach is guaranteed to provide an accurate representation of a model's predictive uncertainty, it is computationally infeasible to carry out exact Bayesian inference on modern machine learning models such as neural networks. Instead, the best researchers can do is to use approximations, meaning that any guarantee that the model's uncertainty will be accurately represented is lost.

Practical Considerations in Using Uncertainty Quantification

Uncertainty quantification methods for machine learning are a powerful tool for making modern machine learning systems more reliable. While no existing approach is a silver bullet and each approach has distinct practical shortcomings, research has shown that methods specifically designed to improve the ability of modern machine learning systems to quantify their uncertainty—such as the approaches described above—succeed at doing so in *most* settings. These methods therefore often serve as "add-ons" to standard training routines. They can be custom-designed to meet the specific challenges of a given prediction task or deployment setting and can add an additional safety layer to deployed systems.

Considering human-computer interaction is crucial for making effective use of uncertainty quantification methods. For example, being able to interpret a model's uncertainty estimates, determining the level of uncertainty in machine learning systems that human operators are comfortable with, and understanding when and why a system's uncertainty estimates may be unreliable is extremely important for safety-critical application settings. Choices around the design of user interfaces, data visualizations, and user training can make a big difference in how useful uncertainty estimates are in practice.[8]

Given the limitations of existing approaches to uncertainty quantification, it is essential that the use of uncertainty estimates does not create a false sense of confidence. Systems must be designed to account for the fact that a model displaying high confidence could still be wrong if it has encountered an unknown unknown that goes beyond what it was trained and tested for.

Outlook

There is increasing interest in how uncertainty quantification could be used to mitigate the weaknesses of large language models, such as their tendency to hallucinate. While much past work in the space has focused on image classification or simple tabular datasets, some researchers are beginning to explore what it would look like for chatbots or other language-based systems to "know what they don't know."[9] This research needs to grapple with challenges specific to language generation, such as the fact that there is often no single correct answer. (For instance, correct answers to the question: "What is the capital of France?" could include, "Paris," "It's Paris," or "The capital of France is Paris," each of which requires the language model to make different predictions about which word should come next.)

Due to the fundamental challenges of reliably quantifying uncertainty, we should not expect a perfect solution to be developed for language generation or any other type of machine learning. Just as with the broader challenge of building machine learning systems that can generalize to new contexts, the possibility of distribution shift means that we may never be able to build AI systems that "know what they don't know" with complete certainty.

Nonetheless, research into reliable uncertainty quantification in challenging domains—such as computer vision or reinforcement learning—has made great strides in improving the reliability and robustness of modern machine learning systems over the past few years and will play a crucial role in improving the safety, reliability, and interpretability of large language models in the near future. Over time, uncertainty quantification in machine learning systems is likely to move from being an area of basic research to a practical engineering challenge that can be approached with the different paradigms and methods described in this paper.

Authors

Tim G. J. Rudner is a non-resident AI/ML fellow with CSET and a faculty fellow at New York University.

Helen Toner is the director of strategy and foundational research grants at CSET.

Acknowledgments

For feedback and assistance, we are grateful to Alex Engler, Heather Frase, Margarita Konaev, Larry Lewis, Emelia Probasco, and Thomas Woodside.

Document Identifier: doi: 10.51593/20220013

Endnotes

[1] Neil Band et al., *Benchmarking Bayesian Deep Learning on Diabetic Retinopathy Detection Tasks*, Advances in Neural Information Processing Systems, 2021, https://openreview.net/forum?id=jyd4Lyjr2iB.

[2] We note that, technically, models are trained to achieve a high cross-entropy between the data labels and the predicted probabilities. This metric does discourage the model—to some extent—from being confident and wrong on the training data but does not necessarily lead to well-calibrated predictions.

[3] For simplicity, we only discuss classification problems, where a model predicts class probabilities that make it easy to compute the calibration of a predictive model.

[4] See, for example, this paper on predicting retinal disease (including Table 4 on expected calibration error): Joost van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal, "Uncertainty Estimation Using a Single Deep Deterministic Neural Network," arXiv preprint arXiv:2003.02037 (2020), https://arxiv.org/abs/2003.02037.

[5] The reason for this is that ensemble methods rely on the assumption that by initializing the neural network weights at the beginning of training to different values (and, when possible, training each ensemble component of a different subset of the training data), each trained model will make a different prediction on points that are very different from the training data, resulting in a high standard deviation (i.e., uncertainty) between predictions. However, depending on the model class, the training data, and the data point where the model is asked to make a prediction, it is possible that in fact all ensemble members make similar predictions, hence resulting in low predictive uncertainty.

[6] More precisely, "similar" here means that the data the model is evaluated on must have been generated by exactly the same underlying data-generating process as the training data.

[7] More precisely, Bayesian inference allows us to infer a distribution over some random variable given the data—called the posterior distribution. To compute a posterior distribution, we require an observation model that tells us how likely it is to observe the data given a specific realization of the random variable of interest and a so-called prior distribution over the random variable that reflect our beliefs about the potential values the random variable of interest could take.

[8] See, for example, Malte F. Jung, David Sirkin, Turgut M. Gür, and Martin Steinert, "Displayed Uncertainty Improves Driving Experience and Behavior: The Case of Range Anxiety in an Electric Car," CHI '15: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, April 2015, https://dl.acm.org/doi/10.1145/2702123.2702479; Matthew Kay, Tara Kola, Jessica R. Hullman, and Sean A. Munson, "When (ish) is My Bus?: User-centered Visualizations of Uncertainty in Everyday, Mobile Predictive Systems," CHI '16: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, May 2016, https://dl.acm.org/doi/10.1145/2858036.2858558.

[9] Cf. Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez et al., "Language Models (Mostly) Know What They Know," arXiv preprint arXiv:2207.05221v4 (2022),

https://arxiv.org/pdf/2207.05221.pdf; Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar, “Semantic Uncertainty: Linguistic Invariances For Uncertainty Estimation in Natural Language Generation,” arXiv preprint arXiv:2302.09664v3 (2023), https://arxiv.org/pdf/2302.09664.pdf.