When not to use machine learning: a perspective on potential and limitations

Matthew R. Carbone*
Computational Science Initiative, Brookhaven National Laboratory, Upton, New York 11973
(Dated: October 4, 2022)

The unparalleled success of artificial intelligence (AI) in the technology sector has catalyzed an enormous amount of research in the scientific community. It has proven to be a powerful tool, but as with any rapidly developing field, the deluge of information can be overwhelming, confusing and sometimes misleading. This can make it easy to become lost in the same hype cycles that have historically ended in the periods of scarce funding and depleted expectations known as AI Winters. Furthermore, while the importance of innovative, high-risk research cannot be overstated, it is also imperative to understand the fundamental limits of available techniques, especially in young fields where the rules appear to be constantly rewritten and as the likelihood of application to high-stakes scenarios increases. In this perspective, we highlight the guiding principles of data-driven modeling, how these principles imbue models with almost magical predictive power, and how they also impose limitations on the scope of problems they can address. Particularly, understanding when *not* to use data-driven techniques, such as machine learning, is not something commonly explored, but is just as important as knowing how to apply the techniques properly. We hope that the discussion to follow provides researchers throughout the sciences with a better understanding of when said techniques are appropriate, the pitfalls to watch for, and most importantly, the confidence to leverage the power they can provide.

Keywords: artificial intelligence; computation/computing; machine learning; modeling

I. INTRODUCTION

The story of artificial intelligence (AI) began in the mid 1900's, when computer scientists started considering a simple question: "can machines think?" [1, 2]. Ever since, the mythical goal of achieving true "human-like" AI has framed decades of scientific conversation and has motivated many key breakthroughs, including the backbone of modern machine learning (ML) algorithms: neural networks [3–5]. As the computational machinery required to realize the practical applications of ML would come decades later [6], its full potential would not yet be immediately understood. That time has come, and despite the turbulence of multiple "AI winters" over the past decades [7–9], we are currently living the AI/ML revolution.

Broadly, AI and ML are two related families of methods that fall under the larger "data-driven" umbrella. Built upon well established theory in the statistics and applied mathematics communities [4], modern-day AI and ML is best understood as the intersection of powerful modeling paradigms with "big-data" and bleeding edge hardware (e.g. graphics processing units; GPUs). The general interpretation (though not the only one [10]) is that AI is a superset of ML [11] and consists of techniques that are used to mimic human cognition and decision-making, whereas ML is more focused on the mathematical and numerical approaches. Often, ML is described as the ability of a program to learn a task without being programmed with task-specific heuristics [12]. However, the distinction between AI and ML is not germane to most applications (and many applications use parts of both), hence the blanket term "AI/ML" is used commonly as a substitute for "data-driven" in many contexts. In this perspective, we will focus primarily on supervised ML, though many of the key points to come apply to data-driven approaches in general.

Cruising behind the slipstream created by tremendous success in the technology sector, ML has found wide applicability in the materials, chemical and physical sciences. For example, the discovery, characterization and design of new materials, molecules and nanoparticles [13–24], surrogate models for spectroscopy and other properties [25–28], self-driving laboratories/autonomous experimentation [29–34], and neural network potentials [35–39] have all been powered by ML and related methods. The current state of ML in materials science specifically has also been thoroughly documented in many excellent reviews [15, 40–42] that cover subject matter ranging from applications to computational screening and interpretation. On a related note, for technical details and timely tutorials, we refer those interested readers to Refs. 43 and 44. However, while the scope of ML-relevant problems is huge, not every problem can effectively leverage the power ML provides. Worse still, sometimes ML may seem to be a perfectly reasonable choice only to fail dramatically [45]; such failures can often be traced back to the foundations of any ML tool: the data.

In this perspective, we ask and answer a foundational question which ultimately has *everything* to do with data: when should you not use machine learning? ML is the jackhammer of the applied math world, and is able to channel incredible power provided by the interplay of highly flexible models, large databases and GPU-enabled supercomputers. But you wouldn't use a jackhammer to do brain surgery. At least for the time being,

* mcarbone@bnl.gov

there are classes of problems for which ML is not well-suited [43, 46]. We address this issue not to dissuade researchers from using these methods, but rather to *empower* them to do so correctly, and to avoid wasting valuable time and resources. Understanding the limitations and application spaces of our tools will help us build better ones, and solve larger problems more confidently and with more consistency.

II. THE DEVIL'S IN THE DISTANCE

Newcomers to the field of ML will find themselves immediately buried under an avalanche of enticing algorithms applicable to their scientific problem [47]. Many of these choices are so sophisticated that it is unreasonable to expect any ML non-expert to understand their finer nuances and how/why they can fail. The steep learning curve combined with their intrinsic complexity, mythical "black box" nature and stunning ability to make accurate predictions can make ML appear almost magical. It may come as a surprise then that in spite of said complexities, almost all supervised ML models are paradigmatically the same and are built upon a familiar quantity: distance.

The supervised ML problem is one of minimizing the distance between predicted and true values mapped by an approximate function on the appropriate inputs. A distance can be a proper metric, such as the Euclidean or Manhattan norms, or something less pedestrian, such as a divergence (a distance measure between two distributions) or a cross-entropy loss function. Regardless, the principle is the same: consider un-regularized, supervised ML, where given a source of ground truth F and a ML model f_θ, the goal is to find parameters θ such that the distance between $F(x)$ and $f_\theta(x)$ is as small as possible for all possible x in some use case. While this is only one type of ML, most techniques share this common theme. For example, Deep Q reinforcement learning [48] leverages neural networks to map states (inputs) to decisions (outputs), and unsupervised learning algorithms rely on the same notion of distance to perform clustering and dimensionality reduction that supervised learning techniques use to minimize loss functions. Variational autoencoders [49–51] try not only to minimize reconstruction loss, but simultaneously keep a compressed, latent representation as close to some target distribution as possible (usually for use in generative applications). Numerical optimization is the engine that systematically tunes model parameters θ in gradient-based ML,[1] and its *only* objective is to minimize some measure of distance between ground truth and model predictions.

[1] The classic numerical optimizer is gradient/stochastic gradient descent, with more recently established developments showing systematic improvements in deep learning, e.g. Adam [52].

Additionally, in order to increase the confidence that ML models will be successful for a given task, it helps if the desired function is smooth: i.e. a small change in a feature should ideally correspond to a relatively small change in the target. This idea is more readily defined for regression than for classification, and the data being amenable to gradient-based methods is is not strictly required for ML to be successful. For example, e.g. Random Forests are not generally trained using gradient-based optimizers, but satisfying this requirement will usually help models generalize more effectively. The distance between the features of any two points of data is informed entirely by their numerical vector representation, and while these representations can be intuitive or human-interpretable, they must be mathematically rigorous.

The devil here, so to speak, is that what might appear intuitive to the experimenter may not be to the machine. For example, consider the problem of discovering novel molecules with certain properties. Molecules can be first encoded in string format (e.g. SMILES [53]), and then a numerical latent representation. The structure of this latent space is informed by some target property [16], and because any point in the latent space is just a numeric vector living in a vector space, a distance can be easily defined. This powerful encoding method can be used to "interpolate between molecules" and thus discover new ones that perhaps we haven't previously considered, but it still relies on the principle of distance, both between molecules in the compressed latent space, and their target properties.

Concretely, the length scales for differentiating between data points in the feature space are set by the corresponding targets. Large changes in target values between data points can cause ML models to "focus" on the changes in the input space that caused it, possibly at the expense of failing to capture small changes. This is often referred to as the bias-variance trade-off. Most readers may be familiar with the concept of over-fitting: for instance, essentially any set of observations can be fit exactly by an arbitrarily high-order polynomial, but doing so will produce wildly varying results during inference and be unlikely to have captured anything meaningful about the underlying function. Conversely, a linear fit will only capture the most simple trends to the point of being useless for any nonlinear phenomena. Fig. 1 showcases a common middle ground, where the primary trend of the data is captured by a Gaussian Process [54], and smaller fluctuations are understood as noisy variations around that trend.

Consider a more realistic example: the Materials Project [55] database contains many geometry-relaxed structures, each with different compositions, space groups and local symmetries at 0 Kelvin. Thus, within this database, changes in e.g. the optical properties of these materials is primarily due to the aforementioned structural differences and not due to thermal disorder (i.e. distortions) one would find when running a molec-

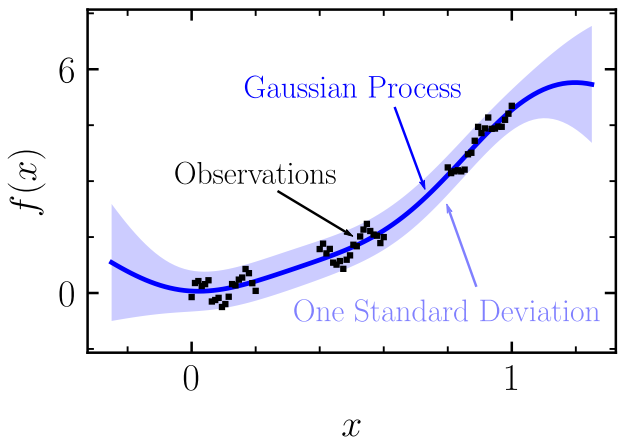

FIG. 1. A Gaussian Process, with a radial basis function kernel, fit to example data $F(x) = 5x^2 + \sin(50x)/3 + \mathcal{N}(\mu = 0, \sigma = 0.1)$. The small, noisy, high-frequency oscillations are not well-fit by a Gaussian Process, as the primary length scale in the data is defined by the distances between the three clusters, not in between them. Sophisticated, non-isotropic kernels could model both trends in principle, but the construction of such kernels often requires significant prior information about the problem, the knowledge of which might make modeling unnecessary.

ular dynamics simulation. A ML model trained on this data could be limited in the sense that changes in certain structural motifs would be well-captured and others would not, necessitating caution when scoping its effective use cases. Conversely, in data-driven modeling, considering the distances between data points in both the input and output spaces, and thus which changes in your features are most contributing to the variance in the targets, can be instrumental in constructing effective training sets targeted to specific applications.

Inverse problems, and cases with extremely low signal-to-noise ratios (SNR), are another case in which the usefulness of ML varies significantly problem-to-problem. ML can only model functions: cases in which each input maps to a unique output. Inverse problems, or those in which a signal is used to resolve its source, are often ill-posed, making it challenging if not impossible to represent them by functions. In these cases, ML is not immediately appropriate, and the problem statement must be refined until a "non-degenerate" subspace is found and can be modeled by a function. Once this space is identified, ML can excel, because it can pick out subtle patterns in this space where heuristics or human intuition may fail to do so, but the developed mapping *must* be a function. This can also be understood in the language of distance: in the inverse problem, an "epsilon-small" change in the input can result in an extremely large change in the output.[2] For example, consider phase retrieval in coherent diffraction imaging [56]. If a detector measures only the intensity of the signal, all phase information is lost by definition, and unless correlations between the intensity and phase exist (which are specific to the sub-problem of interest) and permit such an inverse mapping [57], there is no way to confidently retrieve the phase information. The degree to which this is possible in general depends entirely on the specific system and the data available.

The same issue can be found when the SNR is low or close to 1: no data-driven technique will be useful if targets cannot be distinguished from each other. Relatedly, if the uncertainty during inference on an inverse problem can be *accurately* quantified, then it is possible to use uncertainty-aware models to make predictions with error estimations [58–60]. However, in cases where the inversion is sufficiently ill-posed, the SNR will be so low that different results cannot be resolved. So while it might be possible to make predictions with error bars, they may not be useful.

In summary, it is always an instructive exercise to consider distances between the properties of entries in your database when attempting any data-driven modeling. The key is to fully understand the property of interest, and to choose a database and encoding such that different data points are sufficiently discriminatory with respect to the target property. If this is not possible, no data-driven method will be able to perform effectively. In these cases, utilization of other prior knowledge, more effective data screening or simply another type of technique entirely might be required. We note that there is no harm in simply trying new techniques on e.g. inverse problems where it is unclear how much pertinent information is present in a database that could allow an inverse mapping to be learned. Such possibility, however, should never be confused with certainty.

III. ON THE IMPORTANCE OF DATA DISTRIBUTIONS

To quantify the effectiveness of trained models, evaluation should always conclude with the presentation of evaluation metrics collected on a "testing" subset of the database. In order to avoid training and hyperparameter tuning[3] biases, the testing set should be disjoint from the set of data that was used to train and tune the models (the training and cross-validation databases, respectively). Put even more simply, the rule of thumb is to "blind" yourself from bias as best you can: take a chunk

[2] In the inverse problem, an "input" might be an observable, such as a spectrum, and the output, a structure, such as in the structure refinement problem.

[3] Hyperparameters are un-trained parameters of the model and training procedure, such as the choice of loss function, or the number of neurons in some layer of a neural network.

of data from the full space of data of interest, and not use it, look at it, or otherwise glean any information from it until you are ready to present results on a trained model and completed pipeline. As long as information from the testing set is not used during development, any approach to model tuning is appropriate (though some, such as using cross-validation, are highly recommended). These are the best practices which are critical to any successful ML project, and they are often highlighted in more technical tutorials [43, 44] (and in more technical detail than presented here), but this is not the complete story.

The testing set is almost always discussed as an unbiased sample, a litmus test for how the model will perform on data it hasn't seen before. However, there is another use for the testing set: it should represent the real-world deployment scenario. In other words, the testing set should not only be disjoint from data the model and pipeline have seen before, it should also ideally represent the data on which the model must be performant. It is paramount to keep in mind that these two uses of the testing set are not always the same.

If the training data comes from a different distribution than data from your deployment scenario, it is highly likely the trained model will fail. Human intuition can actually take us far here, as there is a way to easily sanity-check if any two sets of data do *not* come from the same distribution. Simply re-combine them and sample randomly. If you can easily tell the difference (i.e., determine from which distribution a sample originated), your testing set is "out-of-sample" with respect to the training set. This will not always be the case (see e.g. adversarial examples, where human-imperceptible modifications to images can cause otherwise highly accurate ML models to go haywire [61]), but in many scientific problems, it is a critical exercise to perform when planning a research campaign. For example, this can often happen when attempting to train a model on computer-simulated data (which is relatively cheap to obtain) and then deploying it on experimental data (which often requires expensive and time-consuming experiments). Such a use case is common in science, since we tend to have much less data available than in e.g. image recognition problems in computer science (where more labeled data can be simply bought). Indeed, there are some cases where the simulation is sufficiently accurate when compared to experimental measurements (e.g. predicting the space group from pair distribution data [20] or nanoparticle sizes from x-ray absorption spectra; XAS [14]). Other cases will not work nearly as well, such as when comparing experimental and simulated XAS across a diverse crystal structure database [19]. Fig. 2 showcases this possible failure scenario.

While it is not a hard-and-fast rule, one should never take for granted that data-driven models fit on one set of data will perform well on another. On the contrary, ML performs best (and is most powerful) when it is explicitly fit on the same type of data it is expected to perform on. This is a limitation that is often interpreted as weakness;

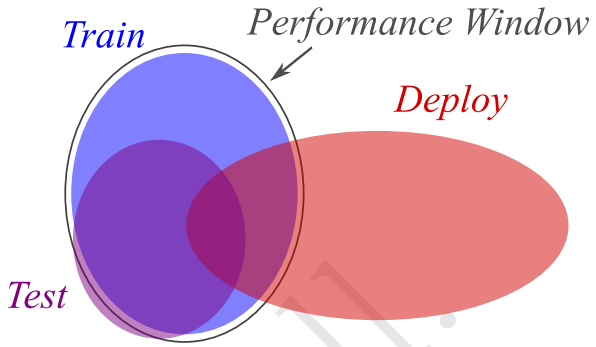

FIG. 2. A possible failure scenario in data-driven modeling: the model is fit to the training (including cross-validation) data, and evaluated on a testing set. The testing set is sampled (but is disjoint) from the same data as the training set, resulting in strong distribution-wise overlap with the training data and likely good performance. The deployment case is sampled from a different database, the distribution of which may only partially overlap with data the model is familiar with.

on the contrary, this is actually a strength. For example, one high-impact example demonstrating this feature is that of neural network potentials [35–39], where the space of possible atomic configurations is kept small (potentials are fit on specific systems), and configurations are revisited throughout long molecular dynamics simulations. Potentials fit on one system are not expected to perform well on another, but they do perform to desired accuracy on the systems they're trained on.

Ensuring proper data selection is not a technical challenge, it is a human one [45]. When considering if ML is the right tool for your problem the real-world deployment scenario must be considered. When at all possible, one should simply fit the model on data from the same distribution as in said deployment. In cases where there is not enough deployment data to fit models on (or to do transfer learning [62]) it is unlikely ML methods will perform well. If the scenario outlined in Fig. 2 is a possibility, sufficient labeled data from the deployment case *must* be available to validate that the model is working properly. If these criteria cannot be satisfied, then there is no feasible way to validate the trained models in the desired use cases. Consequently, if the model's performance cannot be verified, it cannot be used.

IV. THE DATA-DRIVEN "NO FREE LUNCH THEOREM"

The considerable flexibility and information capacity of modern ML models (such as neural networks) comes at a cost. While exceptional at "interpolating" within and close to the "convex hull" defined by the boundaries of the training set, they will often fail in spectacular

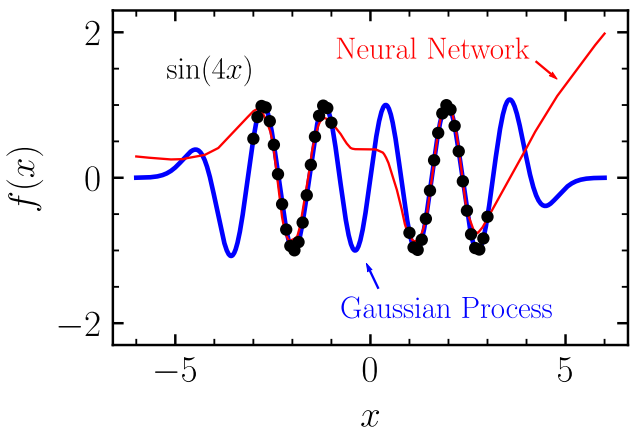

FIG. 3. A simple example of fitting example data (black) with two models: a simple neural network (red) and Gaussian Process with a periodic kernel (blue).

fashion when tasked with predicting outside of this region [36, 46]. This limitation can be partially addressed by the encoding of prior belief, which can take many forms, such as the functional form of the model, correlation information between input features, or boundary behavior. The information content of prior belief can be a game-changer: indeed, even the term "data-driven" can be somewhat misleading [9] (though the description "information-driven", while perhaps more accurate, may be a bit too ambiguous). That said, no data-driven model can make reliable predictions outside of the union of the data and prior information it was trained on. This information-theoretic perspective is tautological, but is often overlooked despite its significant implications: data-driven models generalize, they do not extrapolate with any reliability.

It is important to keep in mind that there is a difference between some trained ML model and an overhead algorithm that is operating for a particular use case which might be using one or more trained models. Often, these algorithms will involve a re-training step in which the inference or decision-making model is continuously updated. For example, reinforcement learning involves an "outer loop" in which the environment is probed and feedback acquired through a reward function, decisions are made, and the decision maker refined. Gaussian Processes and ensemble methods are excellent choices for sampling new data because they naturally quantify uncertainty. Data can be sampled where uncertainty, and thus the likelihood of out-of-sample data, is high. Computer scientists already have a name for this: active learning. This can help the user understand where the model is predicting outside of it's information-theoretic interpolation window, and shore up the model's weaknesses by adding new data to the training set. Another way to interpret active learning is that the algorithm itself is detecting when it is extrapolating, and actively expanding its interpolation window to compensate.

To demonstrate the point, consider the apparently simple 1-dimensional example in Fig. 3, where a neural network and Gaussian Process are tasked with modeling the function $F(x) = \sin(4x)$ with minimal observations. The neural network was trained only to minimize the mean squared error loss function on the provided data. Thus, it ends up fitting the data quite well around where data exists, but fails completely when far away from those points. These failures are essentially random and unpredictable, and are due to the particular state of the neural network's weights. Even "within" the convex hull of the training data, in the region $x \in [-1, 1]$, the neural network does not perform, because the optimizer has no incentive to focus on that region (due to lack of ground truth information). On the other hand, the Gaussian Process is trained using kernels that explicitly encode correlation lengths, limiting the possible set of interpolating functions in the region $x \in [-1, 1]$. This ultimately results in a much better fit, consistent with the learned length scale of the kernel and the data used to fit it. That said, any model that operates on the Bayesian paradigm of starting with a prior belief which is then updated to a posterior when fit on data will revert back to the mean when sufficiently outside of the space of the data it is fit on; this is clearly observed here (the mean of the prior is 0).

One other important observation is that despite the natural human conclusion that the data is likely periodic (which we reach by simply looking at the black markers in Fig. 3), the neural network cannot intuit this without prior knowledge. Thousands more data points could be provided, spanning a much larger range in x, but no amount of data will encode periodicity in a way that a data-driven model understands. On the other hand, when encoded as a prior belief, either in the form of a kernel or perhaps even through explicit selection of a periodic function, only a handful of data points are necessary to completely characterize it. Expecting the model to understand periodicity without explicitly imbuing it with said information is akin to information-theoretic extrapolation: a non-starter.

Unfortunately, most modeling problems of interest are not quite as simple as the 1-dimensional example in Fig. 3. Often, they have much higher-dimensional inputs and outputs, and display complicated non-linear behavior. It is much harder to visualize and interpret results in these situations (though tools are available, such as dimensionality reduction, e.g. Principal Component Analysis), and intuition derived from trial and error is usually the go-to method for understanding when and why the model is not performing well. At the least, care should be taken when formulating data-driven solutions to ensure that the information used to fit the models is carefully thought through. If it is possible the deployment scenario will include out-of-sample data, uncertainty quantification and active learning should be considered. Most of all, the way that humans understand and process data is very different from the way ML algorithms do, and that

should never be overlooked.

V. OUTLOOK: AVOIDING THE NEXT AI WINTER

AI/ML has been transformative in our society over the past decade. Especially in the technology sector, it has been applied with unnervingly surgical accuracy in targeted advertising, image recognition and neural translation, just to name a few examples. In these problem spaces, AI/ML is massively successful because they leverage its greatest strengths: the ability to process huge databases, and complex pattern recognition. Naturally, the scientific community has taken note and applied AI/ML to great effect in research acceleration and discovery. However, these techniques are not a cure-all, and cannot be applied to every problem. Without a doubt, we should keep pushing the boundaries of how we can apply AI/ML in science, but expectations should be kept appropriately measured.

The original AI Winters were caused by outrageously inflated expectations, spurred on by the promise of true AI, and while the actual winter always returns, it is hard to say if another *AI* Winter is on the horizon [8, 63]. In retrospect, the original idea of creating a synthetic autonomous thinker akin to a human was incredibly arrogant. After all, we're competing with millions of years of evolutionary instinct and development, including the most complex black box AI/ML algorithm we know of: the human brain. We would pose a simple question: why compete when we can collaborate?

One day, humanity will likely create sentient AI,[4] but we're not there yet, and for better or for worse, we're not even close. What we *do* have is a wonderful suite of data-driven tools, including those found in the domain of AI/ML, which have the potential to significantly accelerate scientific research and discovery. These tools are meant to empower the experimenter, not to replace them. For the foreseeable future, we must still rely on human researchers to start problems with scientific hypotheses, find appropriate use cases for data-driven tools, and to apply them properly. AI/ML is not magic, and it is of the utmost importance, not only for each individual research project but also to the future of AI/ML in science, that its potential is never taken for granted.

ACKNOWLEDGEMENTS

MRC would like to thank Steven B. Torrisi, Marco Baity-Jesi, Phillip M. Maffettone and Chuntian Cao, for their critical feedback during the writing of this perspective. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Award Numbers FWP PS-030 and DE-SC-0012704.

CONFLICTS OF INTEREST

MRC declares no conflicts of interest.

[1] A. M. Turing, Mind **59**, 433 (1950).
[2] M. I. Jordan and T. M. Mitchell, Science **349**, 255 (2015).
[3] W. S. McCulloch and W. Pitts, Bull Math Biophys. **5**, 115 (1943).
[4] W. S. Sarle, in *Proceedings of the Nineteenth Annual SAS Users Groups International Conference* (SAS Institute, Inc., 1994) p. 1538–1550.
[5] M. Paliwal and U. A. Kumar, Expert Syst. Appl. **36**, 2 (2009).
[6] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, Vol. 4 (Springer, 2006).
[7] D. Crevier, *AI: the tumultuous history of the search for artificial intelligence* (Basic Books, Inc., 1993).
[8] J. Hendler, IEEE Intell. Syst. **23**, 2 (2008).
[9] K. G. Reyes and B. Maruyama, MRS Bull. **44**, 530 (2019).
[10] P. Langley *et al.*, Mach. Learn. **82**, 275 (2011).
[11] A. Holzinger, P. Kieseberg, E. Weippl, and A. M. Tjoa, in *International Cross-Domain Conference for Machine Learning and Knowledge Extraction* (Springer, 2018) pp. 1–8.
[12] J. B. Mitchell, Wiley Interdiscip. Rev. Comput. Mol. Sci. **4**, 468 (2014).
[13] P. Juhás, C. L. Farrow, X. Yang, K. R. Knox, and S. J. Billinge, Acta Crystallogr. A **71**, 562 (2015).
[14] J. Timoshenko, D. Lu, Y. Lin, and A. I. Frenkel, J. Phys. Chem. Lett. **8**, 5091 (2017).
[15] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, and A. Walsh, Nature **559**, 547 (2018).
[16] B. Sanchez-Lengeling and A. Aspuru-Guzik, Science **361**, 360 (2018).
[17] C. W. Coley, W. H. Green, and K. F. Jensen, Acc. Chem. Res. **51**, 1281 (2018).
[18] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik, ACS Cent. Sci. **4**, 268 (2018).
[19] M. R. Carbone, S. Yoo, M. Topsakal, and D. Lu, Phys. Rev. Mater. **3**, 033604 (2019).
[20] C.-H. Liu, Y. Tao, D. Hsu, Q. Du, and S. J. Billinge, Acta Crystallogr. A **75**, 633 (2019).
[21] Y. Zhang, X. He, Z. Chen, Q. Bai, A. M. Nolan, C. A. Roberts, D. Banerjee, T. Matsunaga, Y. Mo, and

[4] For the sake of argument, we will take humanity's survival for granted.

C. Ling, Nat. Comm. **10**, 1 (2019).
[22] S. B. Torrisi, M. R. Carbone, B. A. Rohr, J. H. Montoya, Y. Ha, J. Yano, S. K. Suram, and L. Hung, npj Comput. Mater. **6**, 1 (2020).
[23] R. Mercado, T. Rastemo, E. Lindelöf, G. Klambauer, O. Engkvist, H. Chen, and E. J. Bjerrum, Mach. Learn. Sci. Technol. **2**, 025023 (2021).
[24] V. D. Mouchlis, A. Afantitis, A. Serra, M. Fratello, A. G. Papadiamantis, V. Aidinis, I. Lynch, D. Greco, and G. Melagraki, Int. J. Mol. Sci. **22**, 1676 (2021).
[25] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, in *International conference on machine learning* (PMLR, 2017) pp. 1263–1272.
[26] T. Xie and J. C. Grossman, Phys. Rev. Lett. **120**, 145301 (2018).
[27] M. R. Carbone, M. Topsakal, D. Lu, and S. Yoo, Phys. Rev. Lett. **124**, 156401 (2020).
[28] C. D. Rankine and T. Penfold, J. Chem. Phys. **156**, 164102 (2022).
[29] M. M. Noack, G. S. Doerk, R. Li, M. Fukuto, and K. G. Yager, Sci. Rep. **10**, 1 (2020).
[30] B. P. MacLeod, F. G. Parlane, T. D. Morrissey, F. Häse, L. M. Roch, K. E. Dettelbach, R. Moreira, L. P. Yunker, M. B. Rooney, J. R. Deeth, *et al.*, Sci. Adv. **6**, eaaz8867 (2020).
[31] R. W. Epps, M. S. Bowen, A. A. Volk, K. Abdel-Latif, S. Han, K. G. Reyes, A. Amassian, and M. Abolhasani, Adv. Mater. **32**, 2001626 (2020).
[32] M. M. Noack, P. H. Zwart, D. M. Ushizima, M. Fukuto, K. G. Yager, K. C. Elbert, C. B. Murray, A. Stein, G. S. Doerk, E. H. Tsai, *et al.*, Nat. Rev. Phys. **3**, 685 (2021).
[33] F. Bateni, R. W. Epps, K. Antami, R. Dargis, J. A. Bennett, K. G. Reyes, and M. Abolhasani, Adv. Intell. Syst. , 2200017 (2022).
[34] T. Konstantinova, P. M. Maffettone, B. Ravel, S. I. Campbell, A. M. Barbour, and D. Olds, Digital Discovery , (2022).
[35] J. Behler and M. Parrinello, Phys. Rev. Lett. **98**, 146401 (2007).
[36] J. Behler, J. Chem. Phys. **134**, 074106 (2011).
[37] J. Behler, Phys. Chem. Chem. Phys. **13**, 17930 (2011).
[38] N. Artrith and A. Urban, Comput. Mater. Sci. **114**, 135 (2016).
[39] J. Behler, Chem. Rev. **121**, 10037 (2021).
[40] C. P. Gomes, B. Selman, and J. M. Gregoire, MRS Bull. **44**, 538 (2019).
[41] J. Wei, X. Chu, X.-Y. Sun, K. Xu, H.-X. Deng, J. Chen, Z. Wei, and M. Lei, InfoMat **1**, 338 (2019).
[42] D. Morgan and R. Jacobs, Annu. Rev. Mater. Res. **50**, 71 (2020).
[43] A. Y.-T. Wang, R. J. Murdock, S. K. Kauwe, A. O. Oliynyk, A. Gurlo, J. Brgoch, K. A. Persson, and T. D. Sparks, Chem. Mater. **32**, 4954 (2020).
[44] N. Artrith, K. T. Butler, F.-X. Coudert, S. Han, O. Isayev, A. Jain, and A. Walsh, Nat. Chem. **13**, 505 (2021).
[45] N. Sambasivan, S. Kapania, H. Highfill, D. Akrong, P. Paritosh, and L. M. Aroyo, in *proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (2021) pp. 1–15.
[46] F. Chollet, *Deep learning with Python* (Simon and Schuster, 2021).
[47] T. O. Ayodele, New Advances in Machine Learning **3**, 19 (2010).
[48] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, Nature **518**, 529 (2015).
[49] D. P. Kingma and M. Welling, in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, edited by Y. Bengio and Y. LeCun (2014).
[50] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings* (OpenReview.net, 2017).
[51] C. Miles, M. R. Carbone, E. J. Sturm, D. Lu, A. Weichselbaum, K. Barros, and R. M. Konik, Phys. Rev. B. **104**, 235111 (2021).
[52] D. P. Kingma and J. Ba, arXiv preprint arXiv:1412.6980 (2014).
[53] D. Weininger, J. Chem. Inf. Model. **28**, 31 (1988).
[54] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning* (Cambridge: MIT Press, 2006).
[55] A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, and K. a. Persson, APL Mater. **1**, 011002 (2013).
[56] F. Zhang, B. Chen, G. R. Morrison, J. Vila-Comamala, M. Guizar-Sicairos, and I. K. Robinson, Nat. Comm. **7**, 1 (2016).
[57] L. Wu, S. Yoo, A. F. Suzana, T. A. Assefa, J. Diao, R. J. Harder, W. Cha, and I. K. Robinson, npj Comput. Mater. **7**, 1 (2021).
[58] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, JMLR **15**, 1929 (2014).
[59] A. G. Wilson and P. Izmailov, Advances in neural information processing systems **33**, 4697 (2020).
[60] L. V. Jospin, H. Laga, F. Boussaid, W. Buntine, and M. Bennamoun, IEEE Computational Intelligence Magazine **17**, 29 (2022).
[61] I. J. Goodfellow, J. Shlens, and C. Szegedy, in *International Conference on Learning Representations (ICLR)* (2015).
[62] K. Weiss, T. M. Khoshgoftaar, and D. Wang, J. Big Data **3**, 1 (2016).
[63] L. Floridi, Philos. Technol. **33**, 1 (2020).