# Machine Learning Background for DSECOP Program

Karan Shah

## 0.1 Machine Learning

Machine learning is a form of computational pattern recognition which has had profound impact on multiple fields over the past decade. Advances in GPU hardware coupled with the development of high-level frameworks has led to many different applications in various fields. Some of these applications include autonomous driving [1], protein structure prediction [2], high-energy particle-physics data analysis [3], generative language models [4], and materials discovery [5]. In this section, a brief background on machine learning is given, as well as some specifics relevant to most common models in materials science applications.

### 0.1.1 Introduction

While there is no standard definition of machine learning, it is commonly defined as the study of computer algorithms that improve automatically through experience [6]. Depending on the nature of the model and its relationship with data, machine learning is loosely divided into the following categories.

#### Supervised Learning

Supervised learning refers to the class of problems where some prediction has to be made with respect to a ground truth. A model is trained on training data and its accuracy is tested by comparing its predictions with already known labels. The labels can be continuous real numbers (regression) or discrete states (classification).

Karan Shah
Center for Advanced Systems Understanding, Helmholtz-Zentrum Dresden-Rossendorf, Untermarkt 20, 02826 Görlitz, Germany e-mail: k.shah@hzdr.de

Deep learning [7] is a subset of supervised learning that has gained prominence in the last decade because of its effectiveness in mapping complex datasets.

### Unsupervised Learning and Generative Models

When the dataset consists of only input data without ground truth, unsupervised learning algorithms are used. These algorithms are used for dimensionality reduction, to understand some underlying structure in the dataset, and to generate synthetic data with a similar distribution as the dataset even when such a distribution would be complex, non-trivial, and intractable analytically.

### Reinforcement Learning

In contrast to supervised and unsupervised learning, there is no pre-existing data set in reinforcement learning (RL) problems. The model, referred to as agent for RL problems, has a set of actions that it can perform, which changes the state of the environment. Depending on the state, a score is assigned using the scoring function. The goal of the agent is to maximise this score. In complex environments, the state space may be too large to be enumerated as a discrete matrix. A neural network may be used to learn the mapping between the environment, action, and score. This is called deep reinforcement learning [8].

Machine learning models can be further divided into parametric and non-parametric models. When the model involves a finite set of learning parameters $W$, the model is said to be parametric. These parameters are not set explicitly and learned iteratively through the learning process. Models without a finite set of parameters are non-parametric. In non-parametric models, the set of weights $W$ is generally infinite.

Some examples of machine learning models include *supervised parametric models* (linear regression, neural networks), *unsupervised parametric models* (Gaussian mixture models, generative adversarial networks [9]), *supervised non-parametric models* (Gaussian Process Regression (GPR) [10], Decision Trees (DT) [11], Support Vector Machines (SVM) [12]), and *unsupervised non-parametric models* (k-means algorithm) [13].

Learning models can also be combined into ensembles. Using multiple learning models reduces overfitting and improves generalisability. Examples of ensemble learning techniques are Random Forests [14] and Gradient Boosting (GB) [15].

## 0.1.2 The learning process

Supervised parametric learning problems generally follow a common structure. Given a dataset $D$, a model $M$ with parameters $W$ is to be fitted such that the

cost function $J$ is minimised. The components of the learning process are described broadly.

### Dataset

The dataset $D$ consists of $N$ samples $\mathbf{X}$ and labels $\mathbf{y}$. The *training data* is a subset of the dataset that is used to train the model. Using the known data points $\mathbf{X}_{train}$ and the corresponding labels $\boldsymbol{y}_{train}$, the model is adjusted to find the best possible mapping in the training set. Once trained, a model is only useful if it can be generalised to previously unseen data. This performance is quantified by using the model to predict labels for unobserved data for which true labels are known. The *testing data* is used to measure the performance of the model by comparing predicted labels with true labels.

### Model

The learned mapping between input data $\mathbf{X}$ and labels $\boldsymbol{y}$ is represented by the model $f$. It is a function $f : \mathbf{X} \mapsto \boldsymbol{y}$. This mapping depends on parameters $\mathbf{W}$ of the model. Once trained on the true labels $\boldsymbol{y}^*$, the model is used to make predictions $\boldsymbol{y}$ (inference).

### Loss function

The loss or cost function $J$ is a quantitative measure of the penalty due to error. The cost function changes as the parameters $\mathbf{W}$ are adjusted. The average loss value over the training set is defined as empirical risk. During the training phase, the parameters $\mathbf{W}$ are adjusted to minimise the empirical risk. The difference between loss during training and loss during testing is referred to as generalisability error. Since only training data is observed by the model, the parameters might be tuned such that it performs well on the training set (low empirical risk) but performs badly on unobserved data (high generalisability error). This is called overfitting. When the model does not capture the complexity of the underlying function sufficiently, it is called underfitting. Commonly used loss functions are mean squared error and softmax loss.

### Optimisation scheme

Adjusting parameters in response to the cost function is governed by the optimisation scheme. This is often done by gradient descent algorithms [16]. The parameters are adjusted iteratively according to the gradient of the loss function with respect to the parameters

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \eta \nabla J \ . \tag{1}$$

Here, the learning rate $\eta$ is a parameter that controls the rate of change of model parameters.

### Hyperparameters

The parameters used in the learning process that are set beforehand are called hyperparameters. This is in contrast to the model parameters which are iteratively updated during the learning process. The choice of hyperparameters is important because model performance can be considerably impacted by it. Examples of hyperparameters are the number of iterations in the training process, convergence criteria for the loss function, the learning rate for gradient descent, and the architecture of a neural network.

## 0.1.3 Supervised Learning Models

A brief description of commonly used supervised learning models in materials science is given in the following.

### Linear regression

Linear regression [17] is one of the most used algorithms for regression tasks. Given a dataset $(\mathbf{X}, \boldsymbol{y})$ consisting of labeled data where $(\boldsymbol{x}^{(i)}, y^{*(i)}),\ i = 1, ...N$ with $\boldsymbol{x}^{(i)} \in \mathbf{X}$ and $y^{*(i)} \in \boldsymbol{y}$, regression is used to determine the relationship between dependent variables $y^{*(i)}$ and independent variables $\boldsymbol{x}^{(i)} = \{x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, ..., x_d^{(i)}\}$, where $d$ is the dimension of the data. Given an input $\boldsymbol{x}^{(i)}$, the prediction $y^{(i)}$ is defined as

$$y^{(i)} = \mathbf{W}^T \boldsymbol{x}^{(i)} \ , \tag{2}$$

where $\mathbf{W}$ are the learnt parameters of the model, determined by minimizing the least squares cost function

$$J^{(i)}(\mathbf{W}) = \frac{1}{2} \sum_{i=1}^{N} \| \mathbf{W}^T \boldsymbol{x}^{(i)} - y^{*(i)} \|^2 \ . \tag{3}$$

The analytical solution for optimal value of parameters $W^*$ is given by

$$\mathbf{W}^* = \left( \mathbf{X}^T \mathbf{X} \right)^{-1} \mathbf{X}^T \boldsymbol{y}^* \ . \tag{4}$$

However, the computational complexity for calculating $\mathbf{W}^*$ is $O(N^3)$ which becomes intractable for large data sets with high dimensions. If the number and range of the

parameters is ill-suited for the dataset, the model might reach a sub-optimal state due to overfitting or underfitting. To overcome this, a regularisation term is added to the cost function

$$J(\mathbf{W}) = \frac{1}{2}\sum_{i=1}^{N}\|\mathbf{W}^T\boldsymbol{x}^{(i)} - \boldsymbol{y}^{*(i)}\|^2 \ + \lambda\|\mathbf{W}\|_p \tag{5}$$

with a hyperparameter $\lambda$. When $p = 1$ (i.e., the absolute sum) is used, the model is called Least Absolute Shrinkage and Selection Operator (LASSO) [18]. With $p = 2$ (i.e., the sum of squares), the model is referred to as ridge regression [19]. Sure Independence Screening and Sparsifying Operators (SISSO) [20] is an extension to LASSO with better performance for feature selection. Kernel Ridge Regression (KRR) [21] is a non-parametric realisation of ridge regression. Instead of learned parameters $W$, a kernel function measuring the distance between different data points is used.

Implementing a linear regression model is a simple task and gives interpretable results. However, this technique is not suitable for classification problems and for modelling complex non-linear relationships in datasets that have a high degree of dimensionality.

When classification is needed, logistic regression [22] can be applied. Here, the input data is mapped to discrete labels by applying a sigmoid function on the linear regression model.

## Neural networks

Neural networks (NNs) are a promising approach to regression due to the *universal approximation theorem* [23] that shows that any function can be modeled by a sufficiently sophisticated NN. Deep learning has been widely applied to problems in different domains including image classification [24], linguistics [25], biomedical engineering [26], and cosmology [27]. However, neural networks also have some challenges in determining their hyperparameters and training routines that avoid overfitting.

### *Introduction*

A neural network is essentially a set of nested linear regression functions with non-linear activation functions. The base unit of a neural network is a perceptron or artificial neuron [28]. For data point $\boldsymbol{x}^{(i)}$, the perceptron is defined as

$$f^{(i)} = \sigma(\mathbf{W}\boldsymbol{x}^{(i)} + b)\,, \tag{6}$$

where $\sigma$ is a non linear activation function, $\mathbf{W}$ consists of the weights, and $b$ denotes the bias. When nested together in layers, the model is known as multilayer perceptron or NN. For layer $j$ we have

$$f_j^{(i)} = \sigma_{j-1}\left(b_{j-1} + \mathbf{W}_{j-1}\left(\sigma_{j-2}\left(b_{j-2} + \mathbf{W}_{j-2}\left(\ldots\sigma_1(b_1 + \mathbf{W}_1\boldsymbol{x}^{(i)})\right)\right)\right)\right) . \tag{7}$$

The first layer, that accepts the input samples, is called the input layer, and the final layer is called the output layer. The layers in between are referred to as hidden layers. A layer is defined by the number of its neurons which is often denoted as width. A NN is defined by the number and structure of hidden layers. A network with a large number of hidden layers is called a deep neural network (DNN). This gives rise to the term deep learning [7].

### *Design of a neural network*

The design of neural networks depends on the learning task. The components of a neural network are:

*Activation function:* The choice of activation function affects the performance and output of NNs [29]. Commonly used activation functions are Tanh and ReLU [30].
*Loss function:* The learning objective is quantified by the loss function. Sophisticated loss functions can be designed to achieve multiple objectives [31].
*Optimisation scheme:* Standard gradient descent can be a bottleneck for training DNNs with large datasets [16]. Stochastic gradient descent [32] with mini-batches is used to accelerate the training process. Learning rate and momentum [33] terms are used to reduce overfitting. These parameters can be tuned throughout training with sophisticated optimisation schemes like AdaGrad [34], ADAM [35] and BFGS [36, 37]. The gradients with respect to weights are calculated numerically using backpropagation [16].

### *Architectures*

The operations performed by the neurons can be configured for each layer. The following are some of the most commonly architectures:

*Fully Connected Neural Network (FCN):* Each neuron in a layer of FCNs is connected to every neuron in the next layer. Also called feed forward NNs, FCNs are the earliest and simplest NN architecture [38].
*Convolutional Neural Network (CNN):* CNNs are useful for tasks with two-dimensional or three-dimensional data such as image classification [39], optical character recognition [40], and medical image analysis [41]. Popular CNN models include UNet [42] and AlexNet [43].
*Graph Neural Network (GNN):* Instead of a vector input, GNNs [44] can be used with data points in the form of arbitrary graph structures. This is helpful for many-body problems in cosmology [45] and molecular dynamics [46].
*Recurrent Neural Network (RNN):* RNNs are used for tasks that involve sequential or temporal evolution such as language translation [47], time series predic-

tion [48], and text generation [4]. Long Term-Short Memory (LSTM) [49] is a popular RNN architecture.

*Physics Informed Neural Network (PINN):* PINNs are used to solve partial differential equations. This is done by incorporating initial and boundary conditions in the loss term [50].

NNs are effective at modelling complex non-linear functions and can be generalised well to new data [51]. Shortcomings of NNs are that large training datasets are required for practically useful results, and that the black-box models lack explainability and interpretability [52].

### Gaussian Process Regression

Gaussian Process Regression (GPR) is a powerful non-parametric supervised learning technique with robust uncertainty quantification. Here, a Gaussian random variable $X \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is defined as having the distribution:

$$P(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{d}{2}} |\boldsymbol{\Sigma}|} e^{-\frac{1}{2}\left((\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)} \tag{8}$$

where $\boldsymbol{\mu}$ is the mean, $\boldsymbol{\Sigma}$ is the covariance matrix, and $d$ is the dimension of the data.

A Gaussian process (GP) is a collection of random variables, any finite number of which have a joint Gaussian distribution [10]. It can be thought of as a distribution over functions. By constraining the probability distribution at training points, a learned function can be drawn over the test points. Instead of learning any learning parameters $\mathbf{W}$, inference is done by constructing the appropriate covariance matrix and drawing functions from it. Let $\underline{\boldsymbol{X}}$ be the set of training points, $\mathbf{y}$ be the set of training labels and $\underline{\boldsymbol{X}}_*$ be the set of test points. For a simple noise free case, prediction functions for $y_*$ can be drawn from the distribution

$$\begin{aligned} y_* \sim \mathcal{N}(&\mathbf{K}\left(\underline{\boldsymbol{X}}_*, \underline{\boldsymbol{X}}\right)\mathbf{K}(\underline{\boldsymbol{X}}, \underline{\boldsymbol{X}})^{-1}\mathbf{y}, \\ &\mathbf{K}\left(\underline{\boldsymbol{X}}_*, \underline{\boldsymbol{X}}_*\right) - \mathbf{K}\left(\underline{\boldsymbol{X}}_*, \underline{\boldsymbol{X}}\right)\mathbf{K}(\underline{\boldsymbol{X}}, \underline{\boldsymbol{X}})^{-1}\mathbf{K}\left(\underline{\boldsymbol{X}}, \underline{\boldsymbol{X}}_*\right)\Big) \ , \end{aligned} \tag{9}$$

where $\mathbf{K}_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The kernel function $k$ defines the covariance between any two points in the dataset. Some simple and commonly used kernel functions are squared exponential and periodic kernels [53]. The choice of $k$ is an important design decision and depends on the nature of the dataset. With appropriate kernels and under certain conditions, GPs are equivalent to other machine learning methods such as neural networks [54] and Kernel Ridge Regression [10]. GPs provide in-built uncertainty quantification and are effective for small datasets. However, the inference step involves matrix inversion which scales as $O(N^3)$ for explicit inversion. This makes GPs computationally expensive for larger datasets. Approximate inference to increase the scalability of GPs is an active area of research [55] and tractability

has been greatly increased for sparse matrices with approaches such as Subset of Regressors [56].

### 0.1.4 Further resources and frameworks

Details on machine learning are covered in comprehensive textbooks such as Refs. [57, 58, 59]. A particular focus on deep learning is provided in Refs. [60] and [61], while a historical overview can be found in Ref. [62]. The applications of machine learning in the natural sciences is the central theme in Refs. [63] and [64].

Scikit-learn [65] is a popular python framework for regression and non-deep learning models. PyTorch [66] and TensorFlow [67] are commonly used deep learning frameworks with high level abstractions PyTorch-Lightning [68] and Keras [69], respectively. George [70] and GPyTorch [71] can be used for GPR with support for approximate inference.

## References

1. Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7077–7087. IEEE, June 2021.
2. John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, July 2021.
3. Matthew D. Schwartz. Modern machine learning and particle physics. *Harvard Data Science Review*, March 2021.
4. Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, and Amand others others Askell. Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]*, May 2020.
5. Ghanshyam Pilania. Machine learning in materials science: From explainable predictions to autonomous design. *Computational Materials Science*, 193:110360, June 2021.
6. Tom M. Mitchell. *Machine Learning*. McGraw-Hill Education, New York, 1st edition edition, March 1997.
7. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.
8. Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning, Second Edition: An Introduction*. Bradford Books, Cambridge, Massachusetts, 2nd edition edition, November 2018.
9. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.
10. Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, Mass, 2005.
11. Lior Rokach and Oded Z. Maimon. *Data Mining with Decision Trees: Theory and Applications*. World Scientific, 2008.
12. Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, September 1995.

13. James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.
14. Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001.
15. Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, October 2001.
16. Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv:1609.04747 [cs]*, June 2017.
17. Xin Yan and Xiao Gang Su. *Linear Regression Analysis. Theory and Computing*. WORLD SCIENTIFIC, June 2009.
18. Robert Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.
19. Arthur E. Hoerl and Robert W. Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics : a journal of statistics for the physical, chemical, and engineering sciences*, 12(1):55–67, 1970.
20. Runhai Ouyang, Stefano Curtarolo, Emre Ahmetcik, Matthias Scheffler, and Luca M. Ghiringhelli. SISSO: A compressed-sensing method for identifying the best low-dimensional descriptor in an immensity of offered candidates. *Physical Review Materials*, 2(8):083802, August 2018.
21. Vladimir Vovk. Kernel ridge regression. In Bernhard Schölkopf, Zhiyuan Luo, and Vladimir Vovk, editors, *Empirical Inference*, pages 105–116. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
22. Joseph M. Hilbe. *Logistic Regression Models*. Taylor & Francis, May 2009.
23. Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.
24. Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. September 2020.
25. Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, et al. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, NIPS'17, pages 6000–6010, Red Hook, NY, USA, December 2018. Association for Computational Linguistics.
26. E. Golden Julie, Y. Harold Robinson, and S. M. Jaisakthi. *Handbook of Deep Learning in Biomedical Engineering and Health Informatics. Biomedical Engineering: Techniques and Applications*. Apple Academic Press, July 2021.
27. Amrita Mathuriya, Deborah Bard, Peter Mendygral, Lawrence Meadows, James Arnemann, Lei Shao, Siyu He, Tuomas Karna, Diana Moise, Simon J. Pennycook, et al. CosmoFlow: Using deep learning to learn the universe at scale. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 819–829. IEEE, November 2018.
28. Frank Rosenblatt. The Perceptron: A Perceiving and Recognizing Automaton (Project PARA). Technical Report 85-460-1, Cornell Aeronautical Laboratory, January 1957.
29. Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation Functions: Comparison of trends in Practice and Research for Deep Learning. *arXiv:1811.03378 [cs]*, November 2018.
30. Vinod Nair and Geoffrey E Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814, 2010.
31. Santiago Gonzalez and Risto Miikkulainen. Improved training speed, accuracy, and data utilization through loss function optimization. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, July 2020.

32. Léon Bottou. Large-Scale Machine Learning with Stochastic Gradient Descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of COMPSTAT'2010*, pages 177–186, Heidelberg, 2010. Physica-Verlag HD.
33. David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986.
34. John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.
35. Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. January 2017.
36. Practical Methods of Optimization | Wiley Online Books. https://onlinelibrary.wiley.com/doi/book/10.1002/9781118723203.
37. Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, August 1989.
38. Marvin Minsky and Seymour A. Papert. *Perceptrons. An Introduction to Computational Geometry*. The MIT Press, Cambridge, Mass, expanded, subsequent edition edition, December 2017.
39. Zihang Dai, Hanxiao Liu, Quoc Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. In *Advances in Neural Information Processing Systems 34 preproceedings*. NeurIPS, 2021.
40. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.
41. Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical image analysis. *Annual Review of Biomedical Engineering, Vol 12*, 19(1):221–248, June 2017.
42. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Lecture Notes in Computer Science, pages 234–241, Cham, 2015. Springer International Publishing.
43. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
44. Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, January 2009.
45. Miles Cranmer, Alvaro Sanchez Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases. In *Advances in Neural Information Processing Systems*, volume 33, pages 17429–17442. Curran Associates Inc., Red Hook, 2020.
46. Cheol Woo Park, Mordechai Kornbluth, Jonathan Vandermause, Chris Wolverton, Boris Kozinsky, and Jonathan P. Mailoa. Accurate and scalable graph neural network force field and molecular dynamics with direct force architecture. *npj computational materials*, 7(1):1–9, May 2021.
47. Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.
48. Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1):388–427, January 2021.
49. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997.
50. M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, February 2019.

51. Shuyue Guan and Murray Loew. Analysis of generalizability of deep neural networks based on the complexity of decision boundary. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 101–106. IEEE, December 2020.
52. Keyang Cheng, Ning Wang, and Maozhen Li. Interpretability of Deep Learning: A Survey. In Hongying Meng, Tao Lei, Maozhen Li, Kenli Li, Ning Xiong, and Lipo Wang, editors, *Advances in Natural Computation, Fuzzy Systems and Knowledge Discovery*, Advances in Intelligent Systems and Computing, pages 475–486, Cham, 2021. Springer International Publishing.
53. David Duvenaud. *Automatic Model Construction with Gaussian Processes*. Thesis, University of Cambridge, November 2014.
54. Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep Neural Networks as Gaussian Processes. *arXiv:1711.00165*, February 2018.
55. Malte Kuss, Tobias Pfingsten, Lehel Csato, and Carl E Rasmussen. Approximate Inference for Robust Gaussian Process Regression. Technical Report 136, Max Planck Institute for Biological Cybernetics, March 2005.
56. J. Q. Candela and C. Rasmussen. A Unifying View of Sparse Approximate Gaussian Process Regression. *J. Mach. Learn. Res.*, 2005.
57. Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer, New York, NY, 2nd edition edition, January 2016.
58. Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, Cambridge, MA, illustrated edition edition, August 2012.
59. Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer-Verlag, New York, 2006.
60. Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination press San Francisco, CA, 2015.
61. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning Book*. The MIT Press, Cambridge, Massachusetts, illustrated edition edition, November 2016.
62. Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, January 2015.
63. Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre G.R. Day, Clint Richardson, Charles K. Fisher, and David J. Schwab. A high-bias, low-variance introduction to Machine Learning for physicists. *Physics Reports-review Section of Physics Letters*, 810:1–124, May 2019.
64. Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4):045002, December 2019.
65. G. Varoquaux, L. Buitinck, G. Louppe, O. Grisel, F. Pedregosa, and A. Mueller. Scikit-learn. Machine Learning Without Learning the Machinery. *GetMobile: Mobile Comp. and Comm.*, 19(1):29–33, June 2015.
66. Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
67. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng.

TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
68. William Falcon and The PyTorch Lightning team. PyTorch Lightning, 3 2019.
69. François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.
70. Daniel Foreman-Mackey. George: Gaussian Process regression. *Astrophysics Source Code Library*, page ascl:1511.015, November 2015.
71. Paul Upchurch, Jacob Gardner, Geoff Pleiss, Robert Pless, Noah Snavely, Kavita Bala, and Kilian Weinberger. Deep feature interpolation for image content changes. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 31. IEEE, July 2017.