Deep Learning, Explained:

Fundamentals, Explainability, and Bridgeability to Process-based Modelling

Saman Razavi

Associate Professor, Global Institute for Water Security, School of Environment and Sustainability, University of Saskatchewan, Department of Civil, Geological and Environmental Engineering, Canada
saman.razavi@usask.ca

Abstract

Recent breakthroughs in artificial intelligence (AI), and particularly in deep learning (DL), have created tremendous excitement and opportunities in the earth and environmental sciences communities. To leverage these new 'data-driven' technologies, however, one needs to understand the fundamental concepts that give rise to DL and how they differ from 'process-based', mechanistic modelling. This paper revisits those fundamentals and addresses 10 questions often posed by earth and environmental scientists with the aid of a real-world modelling experiment. The overarching objective is to contribute to a future of AI-assisted earth and environmental sciences where DL models can (1) embrace the typically ignored knowledge base available, (2) function credibly in 'true' out-of-sample prediction, and (3) handle non-stationarity in earth and environmental systems. Comparing and contrasting earth and environmental problems with prominent AI applications, such as playing chess and trading in stock markets, provides critical insights for better directing future research in this field.

Plain Language Summary

Deep learning (DL) is an artificial intelligence (AI) technique that has already served the vast majority, if not all, of everyday society in tasks such as image recognition and language processing through smartphones. The recent unprecedented performance of DL in those tasks has accelerated applications in non-native areas such as earth and environmental sciences where knowledge-based modelling has dominated to date. A major challenge, however, is DL and knowledge-based modelling are rooted in different worldviews towards problem solving. This paper explains the 'whats' and 'whys' of DL from first principles, with an eye on applications since inception in environmental problems. An experiment is run to illustrate the fundamental differences between the two worldviews, and to shed light on some critical, but often ignored, issues DL may face in practice, largely arising from the fact that earth and environmental systems are complex with behaviors changing in ways that are physically explainable but not seen in the period of record due to uncertain factors such as climate change. Such issues must be addressed at the heart of the endeavor to develop DL techniques that embrace the knowledge base available, in anticipation of breakthroughs in an age of big data and computational power.

Keywords

Artificial intelligence, machine learning, deep learning, artificial neural networks, process-based modelling, earth systems, hydrology

Key Points

- DL is rooted in *connectionism, hyper-flexibility,* and *vigorous optimization*, which are alien to conventional knowledge-based modelling.
- A knowledge base is essential to enable credible predictions of *complex, open*, *partially observable*, and *non-stationary systems.*
- Bridging DL and earth and environmental sciences is still embryonic but has great potential in an age of big data and computational power.

Table of Contents

1. The rise of deep learning

The last decade has witnessed a tremendous rise in techniques called ‘deep learning’ (DL), under the umbrella of artificial intelligence (AI) and machine learning (ML), and their unprecedented performance in areas such as computer vision (Krizhevsky et al., 2017), natural language processing (Young et al., 2018), and gaming (Silver et al., 2018). These successes have motivated the application of DL across a wide range of disciplines, including medicine (Hosny et al., 2018), earth sciences (Reichstein et al., 2019), robotics (Torresen, 2018), engineering (Panchal et al., 2019), and finance (Lee et al., 2019). DL owes its exemplary success to the boom in computational power and the emergence of big data sources and associated data storage and sharing technologies.

Earth and environmental sciences appear to be positioned to benefit profoundly from DL, as big data sources on a range of *in situ* and remotely-sensed variables are becoming increasingly available with the advances in sensing technologies (Reichstein et al., 2019). The storage volume of remote sensing data for earth observations is already well beyond dozens of petabytes, with transmission rates exceeding hundreds of terabytes per day. Datasets based on model outputs are rising; for example, the climate assessment dataset provided by the Coupled Model Intercomparison Project Phase 6 may reach 40 petabytes (Eyring et al., 2016). Reanalysis climatic datasets have also grown; for example, NASA’s Modern-Era Retrospective Analysis for Research and Applications version 2 (MERRA-2) is ~400 terabytes (Gelaro et al., 2017). In addition, datasets generated via tens of thousands of citizen science projects are providing large and rich sources of ground-based data.

This potential is shifting the attention of earth and environmental scientists and relevant funding agencies towards ML, as evidenced, for example, by the shift in research work presented at the American Geophysical Union (AGU)’s fall meetings, the largest assembly of earth and environmental scientists with more than 27,000 people in attendance and 25,000 presentations in 2019. The number of ML-related presentations has risen consistently—from 0.2% of total presentations in 2015 to 4.2% in 2020. In particular, this shift has been astonishing in the ‘non-linear geophysics’, ‘earth and space science informatics’, ‘natural hazards’, ‘hydrology’, and ‘seismology’ sub-fields, where 28 (2.1), 18 (5.1), 9 (1.3), 7.5 (1.4), and 6.7% (0.9%) of total presentations, respectively, were related to ML in 2020 (2015).

Recent successful applications of DL techniques to earth and environmental sciences include weather nowcasting and forecasting (Shi et al., 2015; Shi et al., 2017), satellite precipitation bias reduction (Tao et al., 2006), rainfall-runoff modelling (Kratzert et al., 2018; Feng et al., 2020; Ma et al., 2021), rain and snow retrieval from spaceborne sensors (Tang et al., 2018), downscaling hydroclimatic variables (Ducournau and Fablet, 2016), precipitation estimation (Tao et al., 2018; Pan et al., 2019), and surrogate modelling (Gu, et al., 2020; Yu et al., 2020; Vali et al., 2021). Unsuccessful applications, perhaps similar to many other areas, remain largely unreported in the peer-reviewed scientific literature but occasionally appear in other media (e.g., Wexler, 2017; Kolakowski, 2018; Rudin, 2019).

Notably, most DL algorithms, formerly known as artificial neural networks (ANNs), have been around and widely applied in earth and environmental sciences since the early 1990s with the birth of domains such as Hydroinformatics (Abbott, 1991). These applications are documented in reviews by Gardner and Dorling (1998), Maier and Dandy (2000), Krasnopolsky (2007), Maier et al. (2010), Abrahart et al. (2012), Razavi et al. (2012a), Shen (2018), Bergen et al. (2019), and Reichstein et al. (2019). Arguably, however, the uptake of DL to facilitate and advance earth and environmental sciences has not kept pace with data availability and computational power over the past three decades.

But why? The challenges impeding the widespread application of DL to earth and environmental problems to date may be rooted in the fact that convincingly casting those problems, for which an extensive knowledge base is usually available, within the DL framework is often not straightforward. Moreover, the lack of interpretability and explainability of DL has been a major hindrance, as model developers need to be able to make sense of why a model functions the way it does, and to explain that to model users. These challenges can be further complicated in the absence of a solid understanding of the fundamentals of DL and how they differ from theory-driven, mechanistic modelling and prediction. Mechanistic modelling, also called process-based or knowledge-based modelling in this paper, has traditionally been the cornerstone of scientific advancement and policy support.

And why this paper? Motivated by the recent breakthroughs by DL in its original areas of application, namely computer vision and natural language processing, this paper aims to address the persistent challenges facing DL applications in non-native areas related to earth and environmental sciences. With this overarching aim, this paper addresses 10 questions regarding the fundamentals of DL and its explainability and bridgeability to earth and environmental systems modelling:

(1) What is DL and how did it evolve from ANNs?
(2) How can we interpret the internal functioning of DL?
(3) How can the complexity of DL be justified in light of the principle of parsimony?
(4) Why is DL considered superior to other types of ML?
(5) How can DL account for memory and time dependency?
(6) How may DL and process-based models behave differently in out-of-sample prediction?
(7) What can be the often ignored value of domain knowledge in DL?
(8) Why is DL essentially different from process-based modelling?
(9) What are the existing approaches to bridging DL and process-based modelling?
(10) What can we learn from prominent DL applications such as gaming and the stock market?

The structure of this paper is such that it best serves the reader when all sections are followed sequentially. However, an advanced reader could directly refer to a section designated to address a question of interest. Sections 2 through 7 address questions 1 through 6 and sub-sections 8.1 through 8.4 address questions 7 through 10, respectively. A real-world hydrological modelling problem and multiple synthetic functions are used to explain complex concepts via simple examples. The contents of this paper are intended to be accessible to a wide audience from various fields under the umbrella of earth and environmental sciences. However, the views presented mainly arise from the author's data- and theory-driven research background in hydrology and water resources.

2. Back to fundamentals

2.1. Why ML and DL?

ML, and in particular DL, is nowadays concerned with developing machines that improve their own performance in carrying out a given task over time by 'learning' from examples, with minimal human efforts to instruct the machines how to do so (Jordan and Mitchell, 2015). According to Goodfellow et al. (2016), however, the early efforts to generate AI were based on a knowledge base paradigm to instruct machines with a formal set of step-by-step mathematical and if-then rules. Those efforts focused on carrying out tasks that were intellectually difficult for humans but straightforward for computers. Goodfellow et al. (2016) argue such efforts led to no major successes, and the AI of today is about enabling machines to perform tasks that humans perform intuitively and rather easily but have difficulty formally describing how they do so. Examples of such tasks include recognizing faces in a photo or comprehending spoken words.

Not only did state-of-the-art AI divorce from the knowledge base, but it also completely separated from classic data-driven modelling rooted in statistics such as regression. This separation was a response to the need for models that are not constrained by the many assumptions typical statistical models hold. For example, traditional statistical modelling requires a formalization of relationships between variables and assumptions about functional shapes, distributions of variables, and their inter-dependencies, which enables hypothesis testing and the generation of confidence bounds. Conversely, in the ML context the underlying relationships in data may have any complex form, which is typically unknown *a priori*, and the data used may have any size and distributional properties (see Dangeti, 2017, p. 10-11).

Because of these characteristics, ML is deemed suitable to pursue the longstanding ambition to build machines that work with minimal or no human supervision and imposed assumptions. As a result, ML techniques nowadays, and in particular DL, provide flexible tools that can adapt to a wide range of data and applications.

2.2. Evolution of DL and major milestones

It was 1957 when Frank Rosenblatt invented the first algorithm, termed 'perceptron' (Rosenblatt, 1957), which today forms the smallest computational unit of DL. A perceptron, alternatively termed a 'neuron' because of its resemblance to the basic working unit of the brain, is shown in **Figure 1a** and formulated as:

$$y = f\left(\sum_{i=1}^{D} w_i x_i + b\right) \qquad \text{(Eq. 1)}$$

where D is the dimension of input space, ***x*** is the input vector, ***w*** is a set of weights corresponding to the input vector, *b* is bias, and *f* is an 'activation' function. A perceptron has D+1 tunable parameters (i.e., D weights and one bias) and is basically nothing but a multiple linear regression augmented by an output function (*f*), which is non-linear. The form of the activation function was originally a step function, but now a range of monotonic functional forms, such as 'sigmoidal', are used.

The invention of perceptrons created significant excitement in the AI community and beyond. But, it soon became clear that a perceptron would not be able to map input spaces that are not linearly separable, such as the XOR problem (Minsky and Papert, 1969), rendering perceptrons of limited use in real-world applications. The reason for this inability is that the core of the perceptron is a linear regression.

Efforts to overcome this barrier could have followed two different avenues. Perhaps the most intuitive avenue was to employ non-linear regression, by allowing the terms inside the parentheses in Eq. 1 to be of other algebraic forms such as quadratic. However, this was not a viable option in part because the user then would need to specify the form of non-linearity which is not typically known *a priori*, requiring possibly extensive trial-and-error.

The second avenue that led to today's DL was to combine perceptrons both in parallel and in series to create so-called 'multi-layer perceptrons' (MLPs), as shown in **Figure 1b**, with the hope this more complex system could overcome the barrier. An MLP would then have many more tunable parameters than the perceptron. The first layer, also called the first 'hidden' layer, would have D.n_1 weights and n_1 biases, where n_1 is the number of neurons in this layer. Similarly, the second hidden layer would have n_1.n_2. weights and n_2 biases, and the last layer, called the 'output' layer would have n_{d-1}.n_d weights and n_d biases, where n_{d-1} and n_d are the numbers of neurons in the second-to-last and last layers, respectively, and d is the total number of layers. The total number of layers in an MLP and the number of neurons in each layer are 'hyper-parameters', to be specified by users. Also important is the choice of activation functions in each layer. Note that a linear activation function is typically only suitable for the last layer and, in general, any stack of linear layers is effectively equivalent to a single linear layer.

MLPs are a prominent class of 'artificial neural networks' (ANNs), or simply 'neural networks' (NNs), a name reflecting their perceived resemblance to biological neural networks. MLPs, which are sometimes called 'feedforward neural networks' (FNNs), are the building blocks of a range of other ANNs developed later on, including 'autoencoders' (Bourlard and Kamp, 1988), 'recurrent neural networks' (RNNs; Elman, 1990) and its popular variation 'long short-term memory' (LSTM; Hochreiter and Schmidhuber, 1997), convolutional neural networks (CNNs; Lawrence et al., 1997) , and generative adversarial networks (GANs; Goodfellow et al., 2014).

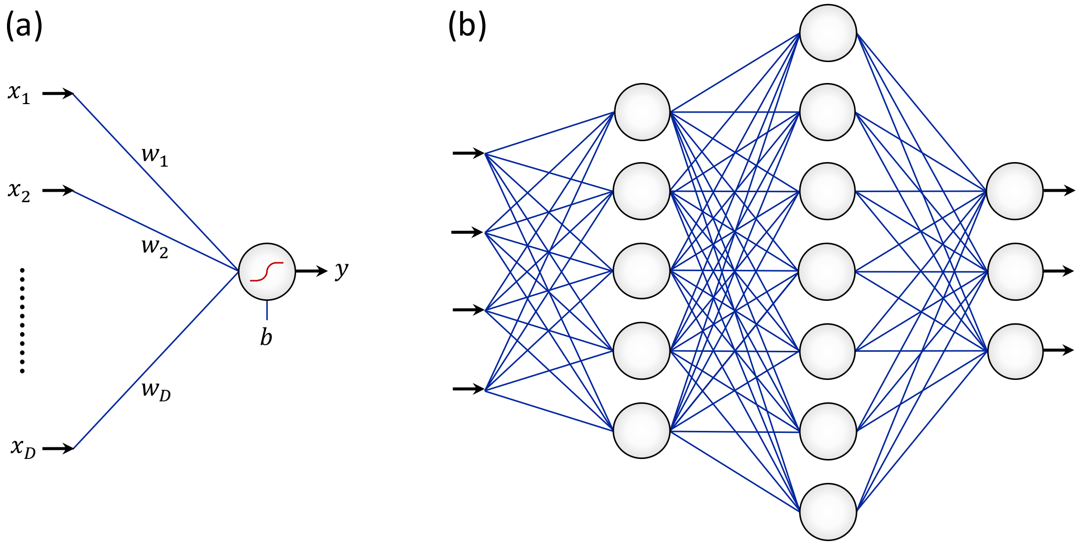

Figure 1. (a) A perceptron and (b) a multi-layer perceptron with four inputs, two hidden layers, and three outputs.

But, MLPs on their own did not go far and the field stagnated for many years because of the absence of an algorithm that could automatically derive from data the network weights and biases—a process referred to as 'training' in the AI community. It took until the mid-1980s when the first 'back-propagation' (BP) algorithm was invented to enable the training of MLPs with any network structure (Rumelhart et al., 1986). This invention marked the beginning of the 'second wave' of popularity of ANNs. BP is essentially an optimization algorithm, based on non-linear programing, that minimizes a loss function representing the goodness-of-fit of predictions to observations, such as the 'sum of squared errors', as follows:

$$F = \sum_{k=1}^{M} \sum_{j=1}^{N} \left(T_j^k - y_j^k\right)^2 \qquad \text{(Eq. 2)}$$

where y_j^k is the output of neuron *j* in the output layer when the network is forced with input data sample *k* and T_j^k is the respective desired target. Also, *M* is the size of training data, and *N* is the number of neurons in the output layer.

Different variations of BP rooted in first- (e.g., gradient descent) or second-order (e.g., Newton's method) optimization, or a combination thereof, now exist; see e.g., the Levenberg-Marquardt algorithm as implemented by Hagan and Menhaj (1994). These algorithms are fundamentally the same as optimization algorithms used nowadays for calibration of process-based models. The only difference is that, in the case of ANNs, and unlike most process-based models, the partial derivatives of the loss function with respect to weights and biases are *analytically* available and obtained through the 'chain rule of differentiation'. More recently, derivative-free and metaheuristic optimization algorithms have shown promise in ANN training (e.g., Dengiz et al., 2009; Rakitianskaia and Engelbrecht, 2009; Razavi and Tolson, 2011), but have yet to become mainstream.

The training of ANNs is an iterative optimization process, where the network parameters are updated after each iteration (called an '*epoch*' in the ANN context), to minimize the loss function. This process can be via 'batch training', where at each epoch the entire batch of training data (i.e., all *M* input-output sets) are used. Alternatively, each epoch can follow 'mini-batch training' based on a subset of training or 'incremental/online training' based on a single training data sample, chosen randomly or otherwise (Hagan et al., 1996). These two approaches, also commonly referred to as 'stochastic gradient descent, are useful when the size of training data is large (Bottou, 1998; Bottou, 2010).

In the late 1980s, after the invention of BP, MLPs were proven to be 'universal approximators' (Hornik et al., 1989). This proof indicated MLPs with only one single-hidden layer that possesses a sigmoidal activation function, and a linear output layer, would be able to approximate any function with any desired level of accuracy provided the number of hidden neurons is sufficient. Since then, the 'universal function approximation theorem' has been the fundamental driver of interest in MLPs across a variety of disciplines and applications.

ANNs started receiving much attention in earth and environmental sciences in the early 1990s. The pioneering applications of ANNs include: Benediktsson et al. (1990), Badran et al. (1991), Stogryn et al. (1994), Bankert (1994), and Cabrera-Mercader and Staelin (1995) in the context of remote sensing of the environment; McCann (1992), Boznar et al. (1993), and Navone and Ceccatto (1994) in the context of atmospheric forecasting; and Kang et al. (1993), Hsu et al. (1995), and Minns and Hall (1996) in the context of hydrology modelling. Perhaps the most prominent and widely used application of ANNs in these fields has been related to the development of PERSIANN, or 'Precipitation Estimation from Remotely Sensed Information using Artificial Neural Networks' (Hsu et al., 1997; Sorooshian et al., 2000; Ashouri et al., 2015), which has been maintained and updated for two decades (accessible at https://chrsdata.eng.uci.edu/).

Despite all of these advances, investments in ANNs and therefore the popularity of ANNs saw a decline in the AI community beginning in the mid-1990s. This was perhaps triggered by failures to fulfill overly ambitious or unrealistic promises by prominent AI scientists (Goodfellow et al., 2016) that brought about somewhat a negative reputation for ANNs (Duerr et al., 2020), as historically observed in ‘AI winters’ (Hendler, 2008). ANNs in earth and environmental sciences, however, remained fairly popular arguably until the mid-2000s. The focus of researchers in these fields was to find novel applications of ANNs across different earth and environmental problems.

It took until early 2010s before the third wave of popularity and interest in ANNs hit, when the field was revived and renamed ‘deep learning’. ‘Depth’ is a recently popularized term and loosely refers to the number of hidden layers in ANNs. A related term is ‘width’, which loosely refers to the number of neurons in hidden layers. Now, a DL model simply refers to an ANN with more than a few hidden layers. All of the recent excitement around ANNs is despite the fact that the structure, formulation, and other properties of MLPs have remained unchanged since their inception, except for some minor modifications. So, one might ask: is DL merely a repackaging and rebranding of what existed before? The next section attempts to answer this question while reviewing the recent milestones.

2.3. Latest developments and rebranding the field

To better understand the recent developments in the field of ANNs, one first needs to know the history around the ‘depth’ concept. MLPs, since their inception, have been used with various numbers of hidden layers, that is with various depths. Most applications, however, remained limited to networks with only one hidden layer until very recently. For example, Razavi et al. (2012a) report that more than 90% of ANNs used for surrogate modelling in water resources literature have only one hidden layer. There was (and perhaps still is) no consensus about a proper network depth, because identifying the optimal network configuration for a given problem and dataset is challenging.

Historically, some researchers favored ANNs with more than one hidden layer, arguing that they require fewer hidden neurons to approximate the same function (see e.g., Tamura and Tateishi, 1997). On the other hand, others asserted that single-hidden-layer ANNs are superior to those with more than one hidden layer with the same level of complexity (see e.g., de Villiers and Barnard, 1993). A discussion on this matter is available in Razavi et al. (2012a, pp 9-10).

Three general reasons historically drove interests towards ANNs with a single hidden layer: (1) the universal function approximation theorem (Hornik et al., 1989), as it provided a compelling argument that such ANNs are fully capable of learning any function; (2) the principle of parsimony, as ANNs with fewer hidden layers are generally deemed less complex and more understandable; and (3) difficulty of training, as ANNs with more hidden layers are more complex to train (see e.g., de Villiers and Barnard, 1993).

So, what recently shifted the status quo towards ANNs with multiple (typically many) hidden layers? Goodfellow et al. (2016) attribute the beginning of this shift to the work of Hinton et al. (2006), where ‘unsupervised learning’ was used to pre-train deep ANNs. They show unsupervised learning could effectively initialize the network’s parameters such that the subsequent training efforts through BP would become more successful. In AI, unsupervised learning refers to a process where a model learns from ‘unlabeled’ examples, which are technically inputs with no associated output. This is as opposed to ‘supervised learning’ where examples (i.e., data points) are ‘labeled’, meaning the output associated with each input is available; this process is called ‘model calibration’ in the context of process-based modelling.

Now, one might ask how unsupervised learning can be of any help in supervised learning. A common method for this purpose uses 'autoencoders', which are a class of ANNs historically used for dimensionality reduction and feature learning (Bourlard and Kamp, 1988). An autoencoder is an MLP, typically trained by BP, with one or more hidden layers that receives input and aims to produce the *same* input as its output. In a typical autoencoder, the middle layer has fewer neurons than the dimension of input, thereby acting as a bottleneck that encodes the input data in a lower dimensional space. The signals in the middle layer preserve the information contained in the inputs, which will be decoded back to the original space in the following layers. Autoencoders can pre-train some layers of a deep ANN such that the weights of those layers capture the main features in input data before passing them to the next layers. After the pre-training phase by unsupervised learning, the ANN needs to be further trained in the conventional supervised manner, using the actual output data and algorithms such as BP.

While the third wave of ANN popularity began by leveraging unsupervised learning to train deep ANNs, Goodfellow et al. (2016) argue the interest has gradually shifted back to the classic learning algorithms, such as BP, even for training deep ANNs. Those classic learning algorithms are now believed to work quite well in the DL context, perhaps due to the emergence of unprecedented computational power. In this regard, a game changer was the introduction of graphics processing units (GPUs) to the ANN community as a powerful tool to massively parallelize and thus expedite training algorithms (Raina et al., 2009). Such computational power has enabled the development of large ANNs, in terms of both depth and width. As such, ANNs with hundreds of millions (e.g., Devlin et al., 2018) or even a trillion parameters (e.g., Rajbhandari et al., 2019) are becoming common.

Such a tremendous revival of the field of ANNs might seem at first surprising to those earth and environmental scientists who have known the field for a long time. This might be due, in part, to the fact that ANNs developed nowadays are fundamentally similar to those developed in the 1990s. Differences, if any in an application, are often in the details. For example, following Glorot et al. (2011), the tendency now is to use the rectified linear unit (ReLU), which is an unbounded function, instead of the standard 'sigmoidal' activation functions (see Eq. 1). The recent boom in data science and cyberinfrastructure and in investments by mega companies, such as Google, in this field might explain this revival, resulting in huge successes in image processing (Krizhevsky et al., 2017) and speech recognition (Young et al., 2018). Perhaps recent rebranding of the field under the title of 'deep learning' might have been in part a marketing strategy (Duerr et al., 2020); while as cited in Schmidhuber (2015a), this term was first introduced by Dechter (1986) to ML and by Aizenberg et al. (2000) to ANNs.

3. Geometrical Interpretation of DL

ANNs have always struggled with explainability and interpretability. Extensive research efforts have endeavored to peer inside the 'black box' of ANNs, via various forms of sensitivity analysis (see Section 3.4 of Razavi et al. (2021) for a review) or geometrical or other types of interpretations (e.g., Benítez et al., 1997; Tickle et al., 1998; Castro et al., 2002; Wilby et al., 2003; Xiang et al., 2005; See et al., 2008;Razavi and Tolson, 2011; Samek & Müller, 2019). Despite all these advances, the issues around explainability and interpretability of ANNs, and of many ML techniques in general, are as relevant today as ever (see Rudin, 2019).

This section utilizes a geometrical interpretation of ANNs to illustrate the internal functioning of ANNs and explain why deeper ANNs can be more powerful than 'shallower' ANNs in learning representations in data. This interpretation is adopted in part from the work of Razavi and Tolson (2011), in which they recast ANNs with respect to a new set of more interpretable variables based on the network functional geometry.

3.1. A perceptron

An MLP is in principle made of a number of perceptrons. Consider an MLP with a single hidden layer with a sigmoidal activation function, as shown **Figure 2a**. Each hidden neuron, e.g., the r^{th} neuron, is a perceptron whose output y_r^1 is multiplied by the weight $w_{1,r}^2$ before entering the output neuron. This hidden neuron, when only having one input x_1, forms a functional relationship such as that shown in **Figure 2b**. This 'sigmoidal unit' can be characterized by three variables: 'slope', 'location', and 'height'. There is one-to-one mapping between these variables and the original network variables, $w_{r,1}^1$, b_r^1, and $w_{1,r}^2$, as shown in the figure. As such, one can directly control the shape of the sigmoidal unit through slope, location, and height, and where needed, map them onto the network's original variables. The benefit of doing so is that, unlike the original variables, the new variables are geometrically interpretable and therefore more intuitive.

Figure 2c shows the geometry of a perceptron with two inputs, x_1 and x_2. In this case, the resulting sigmoidal unit forms a plane that can be characterized by slope, location, and height, plus an additional variable called 'angle' that specifies the direction toward which the sigmoidal unit is facing. This geometry can be extended to perceptrons with three or more (say D) inputs, where the sigmoidal unit becomes a *hyperplane*, characterized by a slope, location, and height and D-1 angles. Full details of this geometrical interpretation, and how it works in practice, are available in Razavi and Tolson (2011). Now let us see in the following how ANNs can approximate any function by putting together a large number of such sigmoidal units.

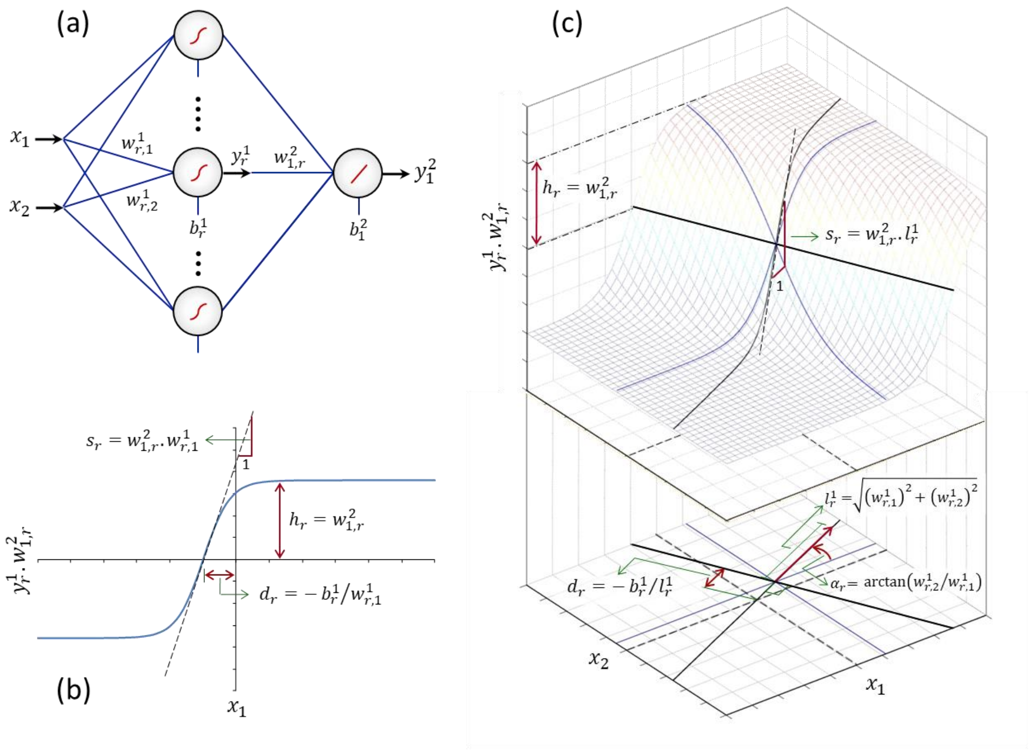

Figure 2. (a) An MLP with a sigmoidal hidden layer and linear output layer. (b) The sigmoidal line formed by the rth hidden neuron when the network has only one input, x_1. (c) The sigmoidal plane formed by the rth hidden neuron when the network has two inputs, x_1 and x_2. A sigmoidal line can be defined by three variables that are related to the original weights and biases: h_r is the 'height' of the tails, s_r is the 'slope' of the tangent line at the inflection point, and d_r is the 'location' of the inflection point with respect to the origin. A sigmoidal plane can be defined based on those three variables as well as α_r, which is the 'angle' of the normal vector perpendicular to the plane. l_r^1 is the length of this vector. This geometry can be extended to MLPs with any number of inputs (see Razavi and Tolson, 2011).

3.2. ANNs with one hidden layer

Single-hidden-layer ANNs are capable of approximating any function by combining, in parallel, as many sigmoidal units as required. For example, suppose the underlying function to approximate is the sine function shown in **Figure 3a**. Three sigmoidal units, with equal heights, equal absolute slopes, and different locations, are required in parallel to represent the features of the function. These three units can be produced by the hidden layer of an ANN and feed into a linear output layer, where they are summed (superimposed) to approximate the sine function, as shown in **Figure 3b**.

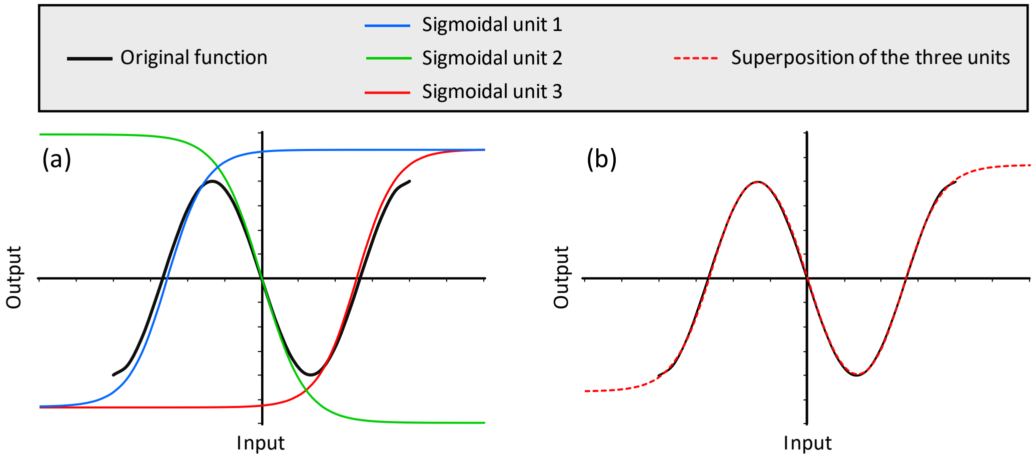

Figure 3. (a) An original sine function and three sigmoidal units, each approximating a part of the sine function. (b) Output of the ANN that superposes the tree sigmoidal units.

For problems with two or more inputs, the function approximation is not as straightforward. For example, suppose the objective in a two-input problem is to approximate the dome-like feature shown in **Figure 4a**. A single-hidden layer ANN with four sigmoidal hidden neurons and one linear output neuron would be able to approximate the dome part of the surface, as shown in **Figure 4b**. This ANN would basically superimpose four sigmoidal units with equal heights, equal slopes, equal locations, but different angles (90° apart). The performance of this ANN, however, is unacceptable, as it creates erroneous features on the tails.

But, can we rectify this issue by using more sigmoidal neurons? **Figure 4c** shows the performance of a network with eight sigmoidal units, all having the same heights, slopes, and locations, but different angles, 45° apart. With more sigmoidal units at work, the performance at the tails is improved, producing less erroneous features. Almost 40 hidden neurons are required, as shown in **Figure 4d**, to generate smooth tails, similar to the original function shown in **Figure 4a**. This example provides a geometrical proof for the universal function approximation theorem of Hornik et al. (1989) because, in principle, any function could be approximated by a combination of such dome-like (i.e., basis) functions. The challenge, however, is that many (possibly an excessively large number of) hidden neurons may be required for a given problem to attain a desired level of approximation accuracy.

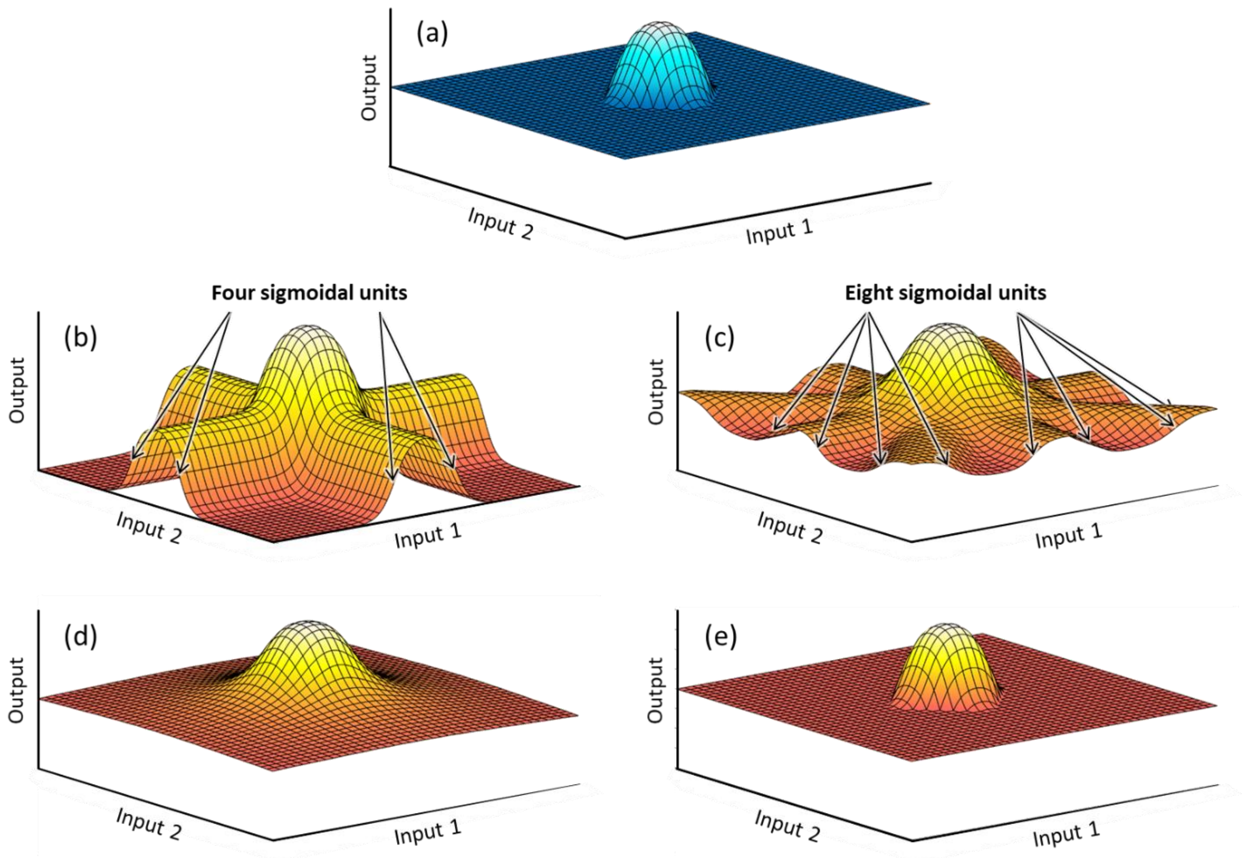

Figure 4. (a) Original dome-like function. Performance of ANNs with (b) four sigmoidal hidden neurons and a linear output neuron, (c) eight sigmoidal hidden neurons and a linear output neuron, (d) 40 sigmoidal hidden neurons and a linear output neuron, and (e) four sigmoidal hidden neurons and a sigmoidal output neuron.

3.3. So, why more than one hidden layer?

As proven by Hornik et al. (1989), and geometrically shown in the example above, ANNs with a sigmoidal hidden layer and a linear output layer are capable of approximating any function with any desired level of accuracy. So, one may wonder about the need to have deeper ANNs. This section attempts to answer this question via an example.

Let us look back at the original function we aimed to approximate in **Figure 4a**. Only four sigmoidal units were required, as seen in **Figure 4b**, to reproduce the dome-like feature at the center. One might ask: Can we stick to these four sigmoidal units and somehow smooth the tails? Yes, all that is needed is a second layer with a nonlinear activation function (e.g., sigmoidal) to deactivate any feature that is under a threshold. In other words, in this process, the geometry formed by the sigmoidal units in the first layer filters through another sigmoidal unit that bounds that geometry. **Figure 4e** shows how adding the second non-linear layer enables the network to reproduce the original function, with only four neurons in the first hidden layer. Similar to single-hidden-layer ANNs, those with two hidden layers can approximate any function by putting the dome-like functions side by side.

In general, shallower ANNs are special cases of deeper ANNs. As shown in the example, deeper ANNs can provide more flexibility while they may require fewer hidden neurons across the network for representation learning. However, the training of deeper ANNs has been historically much more difficult because of the now well-known problem of 'vanishing and exploding' gradients. This problem relates to the fact that the partial derivatives of a loss function (Eq. 2) with respect to weights and biases in first layers, obtained via the chain rule of differentiation, tend to become very small (i.e., close to zero) or very large (i.e., exponentially growing or fluctuating). Improved algorithms along with higher computational power have now eased that difficulty and made possible the training of very deep ANNs (Schmidhuber, 2015b).

Lastly, a related consideration about the proper number of hidden layers is about the fact that, in many problems, only a small part of the input space is active. In other words, some combinations of the different inputs might not occur in reality and therefore the accuracy of the ANN might not matter much in the regions of input space containing those combinations. For example, consider a case similar to one shown in **Figure 4b**, where the corners on the input space do not show up in the data available. A hydrological example is where snowfall and temperature are two inputs to ANNs. Because snowfall would never occur along with high temperature, the respective part of the input space always remains inactive.

4. Relevance of Occam's razor and equifinality?

4.1. Issues with the complexity of ANNs

ANNs are known for their *hyper-flexibility* in fitting data, owing to their enormous degrees of freedom. For example, consider a problem with five inputs and one output. A single-hidden-layer ANN with 10 hidden neurons would have 71 tunable parameters (60 weights and 11 biases), and adding a second 10-neuron hidden layer would result in a network with 181 parameters (160 weights and 21 biases). Compare that with linear or quadratic regression models for the same problem, which would have six or 21 tunable parameters, respectively. Such large degrees of freedom, manifest in large numbers of parameters, encountered in the field of ANNs do not seem consistent with a basic principle in statistical modelling: *Occam's razor*.

Occam's razor, or principle of parsimony, indicates that simpler hypotheses or models should be preferred over more complex ones. In other words, those models that serve the purpose with as few parameters as possible should be chosen. However, many data-driven modellers, in particular in the field of ML, have arguably abandoned Occam's razor. For example, ANN users typically do not try simpler model types such as regression for the problem at hand. And, when using ANNs, they do not necessarily look for the most parsimonious network. Note that some literature proposes systematic approaches to choose a network structure based on growing, pruning, or other strategies (e.g., Reed, 1993; Teoh et al., 2006; Xu et al., 2006). In practice, however, such approaches have been of limited use and most ANN users choose the network structure on an *ad hoc* basis or by trial-and-error (see a survey by Razavi et al., 2012a). Recently, giant ANNs with hundreds of millions of parameters or more have become widespread (Devlin et al., 2018; Rajbhandari et al., 2019).

In addition, *equifinality*, a common and widely discussed issue in process-based modelling (Beven and Freer, 2001; Khatami et al., 2019), is not generally discussed or considered an issue in the context of ANNs. Equifinality concerns the fact that, in most cases, different model structures and parameter values can lead to similar modelling results. In other words, model structure and parameters are not uniquely identifiable from data (Guillaume et al., 2019). This is despite the fact that, loosely speaking, the level of equifinality of ANNs is much larger than other types of models because of their massively parallel nature in producing model outputs.

So, how does DL handle the above issues? The answer is ‘indirectly’, by trying to avoid their undesired implications, which are *overfitting* and *lack of generalizability*. The former refers to a situation where a model fits the noise in the data rather than the underlying function. The latter refers to a case where the model does poorly in ‘out-of-sample prediction’, that is predicting situations unseen in the data used for model training. Various techniques are available in the ANN literature to address these issues, as outlined in the following.

4.2. Leashing the hyper-flexibility of ANNs

Techniques to control the hyper-flexibility of ANNs and to avoid overfitting fall under two general strategies, namely ‘early stopping’ and ‘regularization’. Before reviewing these strategies in this section, let us revisit the common data-splitting approach for calibration and validation of models.

ANNs and traditional, mechanistic models have major differences in terms of calibration and validation. In traditional modelling practices, the available data are commonly divided into ‘calibration’ and ‘validation’ datasets. The former is used to identify the model structure and parameters, while the latter is used to test the model performance in out-of-sample prediction.

In ANN practices, however, the available data are typically divided into three sets, commonly referred to as ‘training’, ‘validation’, and ‘testing’ datasets. Any data chosen for ‘training’ and ‘testing’ in the ANN context are respectively treated like ‘calibration’ and ‘validation’ datasets in the traditional modelling context. The third, ‘validation’ dataset in the ANN context is needed to leash the hyper-flexibility of the network while training. The simultaneous use of ‘training’ and ‘validation’ datasets during ANN training may be best described within the ‘early stopping’ strategy, as follows.

In the ‘early stopping’ strategy, the quality of fit to the ‘validation’ dataset is evaluated after each ‘*epoch’*, that is an optimization iteration trying to minimize the loss function on the ‘training data’ **(see Section 2.2)**. Empirically speaking, as the training error decreases over time, the validation error decreases as well for a while. However, at some particular epoch, the validation error may begin to increase while the training error may keep decreasing **(see Figure 5)**. This epoch is deemed to mark the beginning of overfitting; thus, the user stops the training process. This strategy is therefore called ‘early stopping’ in the sense that the training stops early, before it can further improve the fit to the ‘training’ dataset (for a review, see Prechelt, 1998). When the training process stops, the generalizability of the trained network is assessed via out-of-sample prediction on the ‘testing’ dataset.

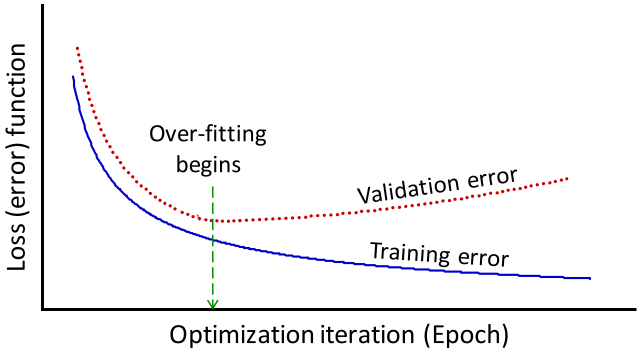

Figure 5. Illustration of ‘early stopping’. The loss function on the ‘training’ dataset generally decreases with more epochs, whereas the loss function on the ‘validation’ dataset decreases early on but begins to increases at some point, marking the commencement of overtraining.

‘Regularization’ is another commonly used strategy to put a leash on the hyper-flexibility of ANNs. Unlike ‘early stopping’, this strategy tries to minimize a ‘regularization function’ during training, to control the ANN flexibility and tailor it to the problem at hand. This strategy has roots in the theory of ‘Tikhonov regularization’ and typically views a more regularized model as one with a smoother response surface (Tikhonov and Arsenin, 1977; Johansen, 1997). A traditional regularization function in the ANN context is the sum of the square of all network parameters (Krogh and Hertz, 1991), based on the notion that, in general, the smaller the parameters of a neuron, the less activated it is. For example, in an extreme case where all parameters of a neuron are zero, that neuron becomes fully inactive and does not contribute a feature to the overall network response. Razavi and Tolson (2011) provide a more efficient regularization function, based on the geometry presented in **Section 3**, where the regularization function is the sum of squares of all of the slopes. This regularization function only targets and removes the unnecessary features, which are unsupported by data, from the overall network response.

But how can one balance the goodness of fit and smoothness of the network response? In practice, this is a bi-objective optimization problem, where one objective is to minimize the error function and the other is to minimize the regularization function. These two objective functions are commonly integrated into one loss function via weighting schemes. **Figure 6** shows how the two objectives compete in a real example. Ideally, one may wish to achieve a performance such as that shown in **Figure 6e**. Doing so is not trivial, however, because in practice the underlying function is unknown, available data are limited, and response surfaces are multi-dimensional and cannot be easily visualized. The Bayesian regulation method developed by MacKay (1992) and extended by Foresee and Hagan (1997) has proven useful to adaptively assign the weights associated with each function during training.

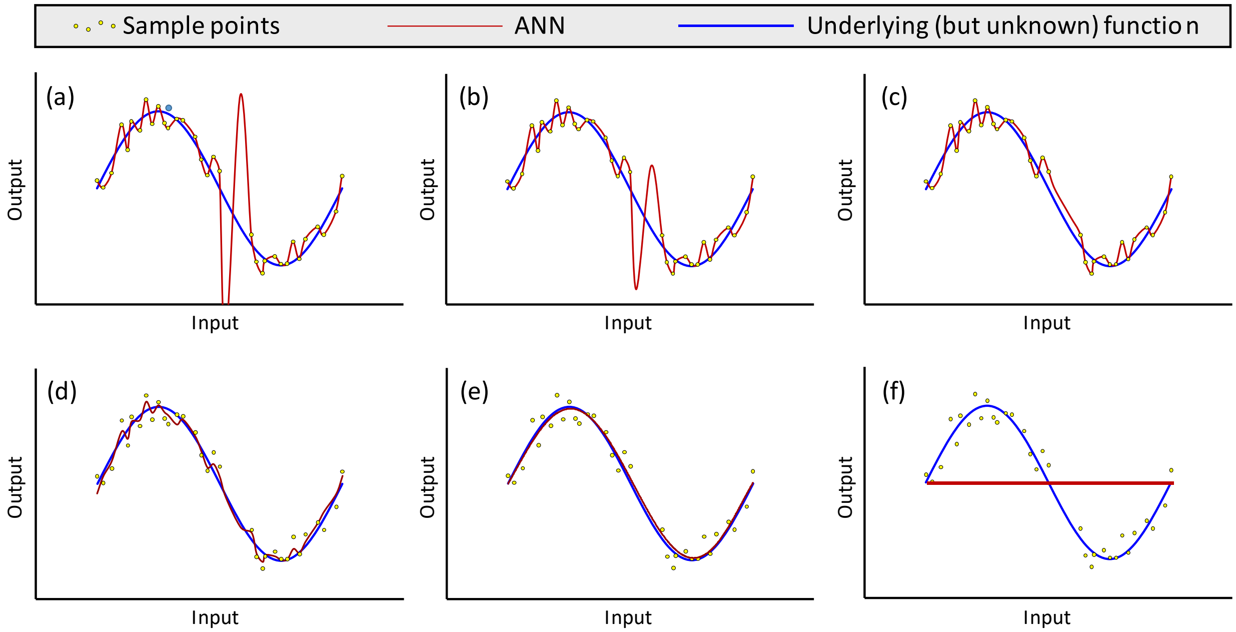

Figure 6. Illustrative example of how regularization works to leash the hyper-flexibility of ANNs. Plot (a) shows an extreme case with no regularization where the ANN overfits data. Plot (b) shows a case where the regularization function is added to the loss function but marginally weighted. Plots (c) through (e) show cases with incremental increases in the weight of the regularization function. Plot (f) shows the other extreme case where the regularization function is dominantly weighted, making the ANN effectively inactive. These plots are based on a real experiment, where the data sample was taken from the underlying sine function shown and polluted with random noise.

A more advanced and recently developed regularization strategy is called 'dropout' (Hinton et al., 2012; Srivastava et al., 2014). 'Dropout' is a heuristic, particularly designed for deep ANNs, that randomly deactivates and then activates different neurons or groups of neurons at each epoch in the course of training. When a part of an ANN is inactivated in this process, the resulting network is called a 'thinned' network. The ultimate prediction after training with dropout is viewed as an approximation of the ensemble average of predictions by many independent ANNs. Basically, the many different thinned networks created throughout the process are assumed to represent ANNs with different configurations and parameters. This heuristic discourages neurons to co-adapt too much and, as such, is believed to avoid overfitting.

5. Fundamental differences from other ML methods

5.1. Local versus distributed representations

Most ML methods, such as those based on kernel functions, are based on 'local representations'. These methods, while forming *connectionist* networks like ANNs, represent each entity (e.g., a training sample point in the input space) via a single processing unit. For example, radial basis functions (Broomhead and Lowe, 1988), Gaussian emulator machines (Kennedy and O'Hagan, 2000), and support vector machines (Vapnik, 1998; Cherkassky and Ma, 2004) may use as many kernels as the number of training samples.

Each kernel typically has a limited radius of influence in the input space, and therefore only responds to inputs located in their local neighborhood.

Conversely, a unique feature of ANNs is their ability to learn through 'distributed representations' (Hinton et al., 1986). They typically represent an entity via collective efforts distributed among multiple processing units (e.g., sigmoidal units). Unlike kernel functions, the sigmoidal units typically have large regions of influence (**see e.g., Figure 2c**) that overlap each other in the input space (**see e.g., Figure 4b**). The former figure shows that a sigmoidal unit influences the entire input space, by dividing it into three zones: lower tail, upper tail, and slope. The latter figure shows how the influences of four such sigmoidal units are superimposed to generate the network response.

5.2. Implications for users

The use of distributed representations has several practical implications. To the author's knowledge, these include:

- ***Transparency:*** The internal functioning of methods based on local representations is more transparent. Local representations are the most straightforward and easy-to-interpret way of learning, whereas distributed representations can be complex, often leading to emergent properties that cannot be easily explained by local representations (Hinton et al., 1986).

- ***Learning difficulty:*** Distributed representations are more difficult and time-consuming to learn. In local representations, the role of each processing unit may be assigned independently of the other units, but in distributed representations, many processing units may be configured together in complex ways to represent a feature in the data.

- ***Network size:*** Distributed representations need much smaller network sizes. In general, the size of the networks based on local representations is directly proportional to the size of the dataset, in most cases with a proportionality constant of one; that is, the number of processing units mirrors the number of training data samples. The size of networks based on distributed representations, however, depends on the complexity of features in the dataset, not its size.

- ***Inexact interpolation or emulation:*** Networks based on distributed representations are generally 'inexact emulators'. This means they do not exactly fit the training samples to represent the features and patterns in the data. This is unlike some other ML methods, such as radial basis functions (Broomhead and Lowe, 1988) and Gaussian emulator machines (Kennedy and O'Hagan, 2000), that are 'exact emulators', perfectly interpolating the training samples. Other inexact emulators include support vector machines (Vapnik, 1998; Cherkassky and Ma, 2004) and multivariate adaptive regression splines (MARS) (Friedman, 1991). Refer to Razavi et al. (2012a, Section 2.6.2) for a discussion on this issue.

In addition, ANNs are essentially multi-output models because they can have as many output neurons as required for a given problem. This means a single ANN can simultaneously predict different variables while accounting for their possible cross-correlations. Many other ML methods are, however, single-output models. For example, in the case of support vector machines, one need to develop two independent models to be able to predict two different variables in a system. Refer to Razavi et al. (2012a, Section 2.6.5) for an extensive discussion on this matter.

6. How to introduce order, time-dependency, and memory

MLPs provide *static* mapping from inputs to outputs. However, many applications require mappings with a formal representation of time evolution and memory. To enable MLPs to do so, two general sets of tools, and combinations thereof, have been used in the literature: (1) tapped delay lines and (2) recurrent connections. These tools are explained in the following.

6.1. Tapped delay lines

A tapped delay line (TDL) consists of a certain number of time delay operators arranged in an incremental order **(Figure 7a)**. TDLs can be installed on any internal connection weights of MLPs to represent time explicitly. The resulting ANN shown in **Figure 7b**, commonly referred to as a ‘time delay neural network’ (TDNN; Waibel et al., 1989), has been widely used in a range of time-series processing applications. As such, TDNNs possess a *static memory with an adjustable length*. This length can be viewed as a hyperparameter to be tuned during training, along with network structural properties such as the numbers of layers and neurons in each layer.

Adding TDLs to an MLP significantly increases the number of tunable parameters. For example, a standard MLP with three inputs and 10 neurons in the first hidden layer would have 30 weights in that layer, while adding TDLs with a length of five to the inputs would result in an additional 50 weights (80 in total) to be trained.

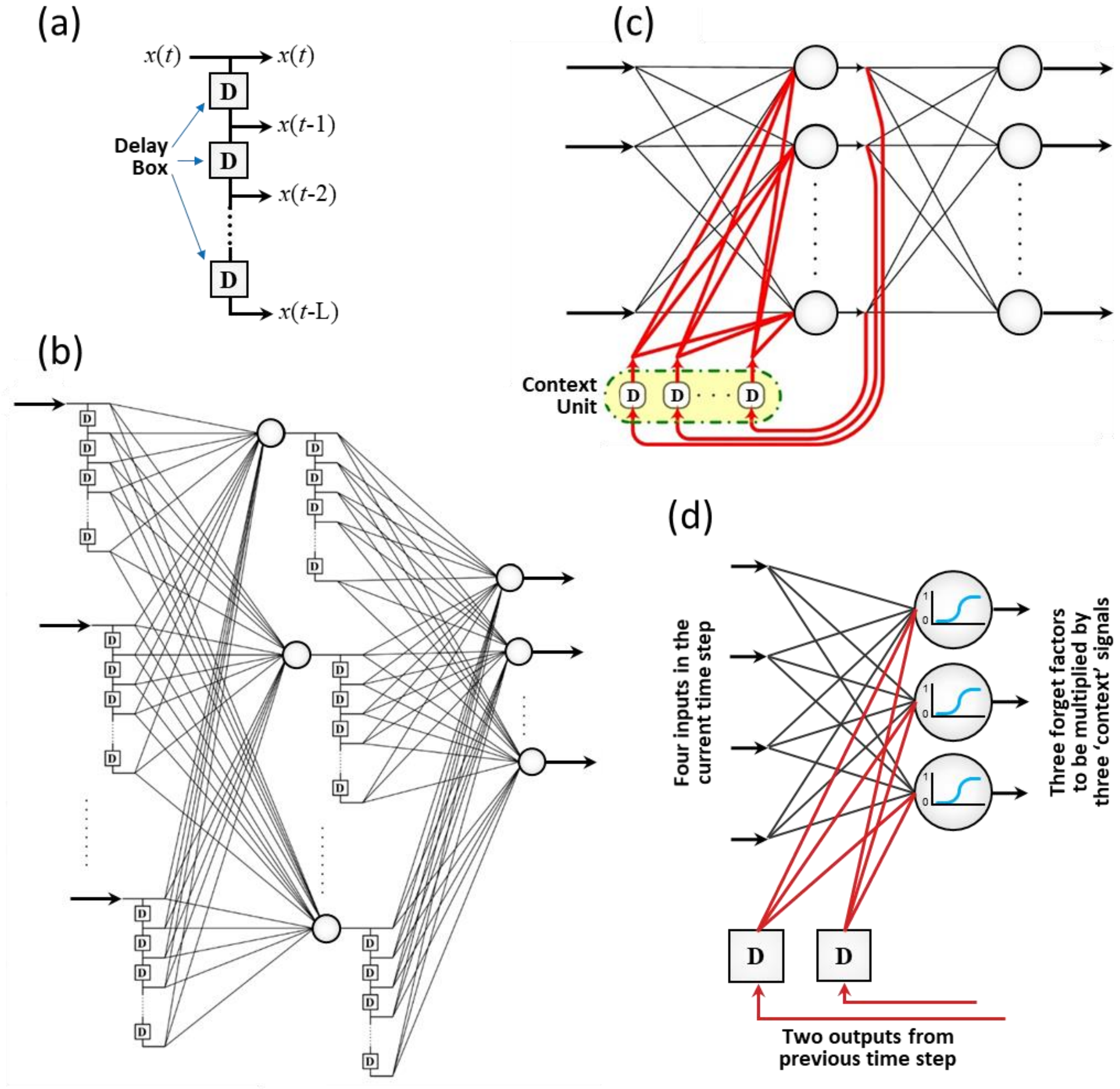

Figure 7. (a) A tapped delay line (TDL), receiving the scalar *x*(*t*) at each time step *t* and outputting the vector [*x*(*t*), …, *x*(*t*-*L*)], where *L* is the length of the TDL. (b) A time delay neural network (TDNN) with one hidden layer and TDLs installed on the input and hidden layers. (c) A recurrent neural network (RNN) with one hidden layer and recurrent connections from the hidden neurons to themselves. In case of long short-term memory (LSTM) networks, the context unit contains three 'gate layers' that adjust the properties of the network's memory. (d) A gate layer of an LSTM with four inputs, two outputs, three 'context' signals that evolve through time steps.

6.2. Recurrent connections

TDLs, as described in **Section 6.1**, explicitly represent time with a memory unit of limited length. Unlike TDLs, recurrent connections, first introduced by Jordan (1986), enable ANNs to account for time evolution based on an implicit memory concept, which is theoretically of unlimited length and is highly context dependent (Elman, 1990). Recurrent connections receive the outputs of a layer at every time step and feed them back to the same or some other layer in the next time step. Technically, they do so via a 'context unit' that stores those outputs in a set of delay boxes **(Figure 7c)**. Recurrent connections can be installed on one or more layers (e.g., Jordan, 1986; Elman, 1990) or locally on some select neurons (e.g., Frasconi et al., 1992).

An MLP enabled with recurrent connections is commonly called a 'recurrent neural network' (RNN). An RNN can possess many more tunable parameters compared to an MLP with the same number of layers and neurons. Using the example given in **Section 6.1**, an MLP with three inputs and 10 neurons in the first hidden layer would have 30 weights in that layer, whereas adding recurrent connections to that layer **(e.g., Figure 7c)** would add 100 more weights (130 in total) to that layer.

Unlike TDNNs that possess a short-term memory, RNNs in theory can represent long-term dependencies in the input sequence as well. In practice, however, recurrent connections have difficulty representing long-term memory because they can easily get dominated by short-term memory. In other words, even very small features arising from short-term dependencies tend to mask features arising from long-term dependencies. In addition, RNNs are prone to the 'exploding and vanishing' gradients problem in their training (Bengio et al., 1994). This is because RNNs, even with a single hidden layer, are in principle deep networks implicitly possessing an infinite number of recursive layers.

6.3. Gate layers to forget or preserve over time

To explicitly account for and balance both short- and long-term dependencies in input sequences, Hochreiter and Schmidhuber (1997) introduced a new type of RNNs, called 'long short-term memory' (LSTM). They extended and further parametrized the 'context' (also called 'cell') such that the network can more explicitly control what information to hold over time and what to forget. The LSTM's context unit modulates not only the outputs in the previous time step but also the inputs to the network in the current time step. It does so via three independent layers of neurons arranged in the so-called 'forget gate', 'input gate', and 'output gate' layers. The neurons of each 'gate layer' as shown in **Figure 7d**, at each time step, receive recurrent connections as well as the new input to the network, and generate their response between *zero* and *one* via using a logistic function. These responses are then multiplied by their respective signals flowing through the context, which means a value of zero would kill a signal whereas a value of one would fully preserve it. Due to the additional weights and biases in the gate layers, an LSTM typically has many more tunable parameters than a conventional RNN.

LSTMs are now perhaps the most popular and widely used type of ANNs with memory. However, LSTMs took a long time (more than a decade) to become known and mainstream, particularly beyond their core computer science community. Their widespread application nowadays owes to recently developed software tools such as Python's *TensorFlow* that efficiently implement variations of LSTMs for a range of problems.

6.4. Training considerations when the order of data matters

The training of memory-enabled ANNs, such as TDNNs, RNNs, and LSTMs is different from that of standard ANNs in terms of the way time-ordered data are presented to the network. To train standard ANNs, the data entries can be presented in any order even randomly, for example through stochastic gradient descent (Bottou, 2010). In memory-enabled ANNs, however, the data entries should be presented in order of occurrence so that the structure of the time dependency is preserved. While this point might seem trivial, it requires careful attention in practical applications.

Another point to consider in the training of memory-enabled ANNs is that all data entries are typically viewed to have equal importance, regardless of their location in the sequence. When used in an online operational forecast, however, the 'forgetting factor' approach can be used to discount older samples. This approach allows the network to adapt to non-stationary environments, where more recent data are more representative of the underlying processes than older data (Razavi and Araghinejad, 2009).

Lastly, elements of TDNNs and RNNs can be combined in a variety of ways. A well-known combination is 'time-delay recurrent neural networks' developed by Kim (1998) and used in various applications such as long-term precipitation forecasting in Karamouz et al. (2008); see Razavi and Karamouz (2007) for a comparison of MLP, TDNN, RNN, and TDRNN in the context of flood forecasting. While such combinations may show improved modelling power compared to other ML or statistical methods, the attribution of memory gains to the different elements can arguably be challenging, if possible at all.

7. ML versus process-based modelling – An experiment

ML has been extensively used to model systems for which process-based models are also available. Process-based models are based on the physics governing the underlying processes and are therefore typically evaluated based on both their physical realism and goodness of fit to data. ML, however, does not do much, if anything, with the underlying physics while reportedly doing a superior job in fitting data, even in out-of-sample prediction. A fairly large body of literature benchmarks ML techniques, particularly ANNs, against process-based models. Examples of such comparisons (directly or indirectly) in the context of hydrologic modelling include Hsu et al. (1995), Tokar and Markus (2000), Wilby et. (2003), Kratzert et al. (2018), Kratzert et al. (2019), Feng et al. (2020), and Ma et al. (2021). Some studies, such as Wilby et al. (2003), also detected correlations between the weights of an ANN and state variables of a process-based hydrologic model as a way to verify that their ANN can capture the underlying processes in a hydrologic system.

This section provides an experiment that runs and compares both types of models for the same problem and walks the reader through all of the steps involved. In particular, the processes around calibration and validation, role of physics, and interpretations of out-of-sample prediction are discussed. This experiment is performed in the context of hydrologic modelling, which has seen tremendous progress over the years with respect to both ML and process-based modelling.

7.1. Data and models

The case study used aims to model the hydrologic system of the Oldman River watershed in Alberta, Canada. This watershed has an area of 1434.73 km^2 at Waldron's Corner with a long-term average temperature of 2.2 °C. On average, this watershed receives 611 mm of precipitation (rainfall + snowfall) annually and generates 11.7 m^3/s of river flow. **Figure 8** shows the 30-year long daily time series data used. The first 22 years were used for model 'calibration' (i.e., the 'seen' data in model development) and the last eight years for model 'validation' (i.e., the 'unseen' data in model development). The first three months of the calibration period were used for model spin-up. In the case of DL, the calibration period was further broken into 'training' (17 years) and 'testing' (5 years) periods, the latter for early stopping of the training process to avoid overfitting. Note that, as explained in **Section 4.2**, the naming convention in the DL context for the 'validation' and 'testing' periods is often the other way around.

To model this system, an LSTM configuration was chosen here as a state-of-the art DL model that accounts for time dependency and memory. The inputs to the LSTM model are daily precipitation and temperature **(Figures 8a and d)** and the output is the concurrent flow **(Figure 8e)**. The LSTM structure was rather arbitrarily chosen to have one hidden layer with five neurons, resulting in 166 calibration parameters. For benchmarking purposes, a classic hydrologic model called HBV (Lindström et al., 1997), as implemented in HBV-SASK (Razavi et al., 2019), was used. HBV-SASK is based on a conceptualization of physical principles governing the water movement in a watershed using 12 calibration parameters. Each of these parameters has a physical interpretation and a physically justified feasible range (see Figure 9 and Table 2 of Razavi et al., 2019). Full detail (including data) of this Oldman River watershed case study, which has been developed for educational purposes, is available in Razavi et al. (2019).

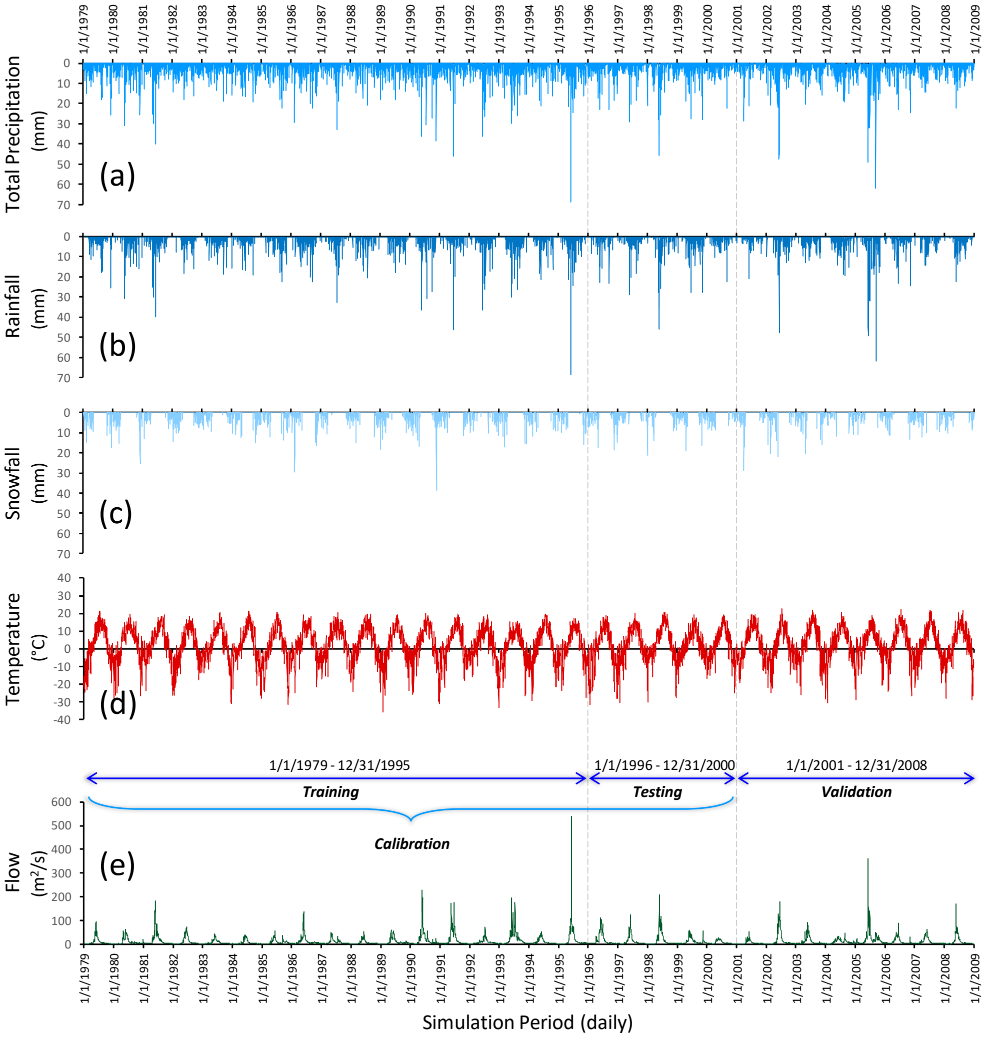

Figure 8. Dataset used for the modelling experiment with ML and mechanistic modelling. (a) Measured precipitation time series (rainfall + snowfall). (b) Estimated rainfall time series (precipitation when

temperature ≥ 0 °C). (c) Estimated snowfall time series (precipitation when temperature < 0 °C). (d) Measured temperature time series. (e) Measured river flow time series. The training period was used for LSTM training, while the testing period was used for early stopping. The calibration (training + testing) period was used for HBV calibration. The validation period was used to evaluate the performance of both LSTM and HBV in out-of-sample prediction.

7.2. Model performance in calibration

The model calibration problem was cast as an optimization problem that tries to maximize the goodness of fit to data by tuning the model parameters, with the Nash-Sutcliffe efficiency (NSE; Nash and Sutcliffe, 1970) as the objective function. NSE is essentially a normalized version of mean squared errors computed as 1-[VAR(errors)/VAR(observations)]. As such, an NSE of one indicates a perfect fit, and an NSE of zero indicates the model prediction is not any better than the average of observations. As a rule of thumb, hydrologists often call an NSE of 0.7 and higher an acceptable fit.

The LSTM model was calibrated using BP with the early-stopping strategy to avoid overfitting. In each epoch, the training period data were used to update the network parameters, while the testing period data were used to detect possible overfitting. Five independent replicates of LSTM calibration (with different initial random seeds) were conducted to account for possible variability of model performance. **Figure 9a** shows the training results of the five replicates compared to a case where the training would not have stopped. As expected, the LSTM performance keeps improving in training, whereas in testing it begins to significantly degrade at some point. The objective function in training came very close to one after many more epochs but with very poor performance in testing (not shown).

The HBV-SASK model was calibrated by a multi-start Newton-type optimization algorithm. Similar to LSTM, five independent replicates of HBV-SASK calibration were run. **Figure 9b** compares the performance of HBV-SASK with that of LSTM in calibration. At this point, only check the performance of the 'standard' LSTM model in calibration. The figure shows all five replicates of LSTM outperform those of HBV-SASK. Note that the calibration performance of HBV-SASK shown herein is almost the best the author has achieved so far for this watershed. Based on these results, the superiority of LSTM over HBV-SASK in calibration is quite significant from a hydrologic modeling point of view. The performance of the two models in validation is discussed in **Section 7.4**, but before that let us discuss what information the two contained prior to calibration.

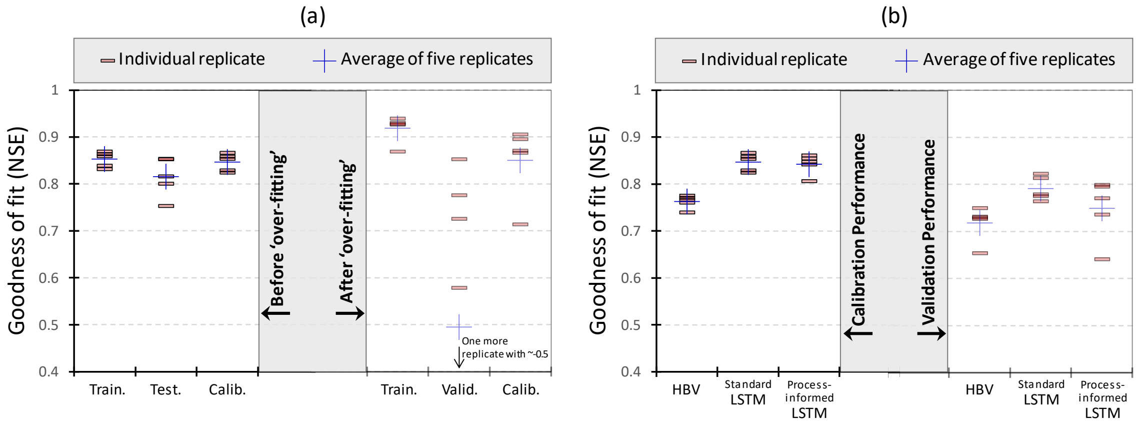

Figure 9. (a) The performance of LSTM in training, testing, and calibration (training + testing) periods before and after 'overfitting'. Training of each replicate was stopped once overtraining began at epoch numbers ranging from 30 to 110 (left panel). Then, each replicate continued to complete 250 epochs in total to merely evaluate the impact of overfitting (right panel). (b) A comparison of LSTM and HBV in out-of-sample prediction. Standard LSTM and process-informed LSTM are discussed in **Sections 7.4 and 7.5**, respectively.

7.3. What about *a priori* information encoded in models?

At this point, let us step back and investigate what we have achieved in terms of learning from data for both the LSTM and HBV-SASK models. The development of the LSTM model was not based on any *a priori* knowledge of how a watershed system works and the governing physical principles. As such, the model learned everything from scratch merely using examples from data. Basically, the model started with a fully randomized internal configuration controlled by a large number (i.e., 186) of parameters and then tuned those parameters to adapt the internal functioning of LSTM to the underlying real-world system represented in the data. **Figure 10a** shows the LSTM performance of arbitrarily chosen replicates before and after calibration. The model response to inputs before calibration seems to be completely random but, after calibration, the model response has learned to closely follow the underlying system response.

Unlike LSTM, HBV-SASK encodes the expert knowledge available in the field of hydrology. This model is a collection of conservation of mass equations and process parametrizations that represent how hydrologists conceptualize the way a watershed works. This 'physically based' modelling structure is presumably able to emulate the behavior of any watershed by tuning only 12 parameters. **Figure 10b** shows how the model performs before calibration, with parameter values chosen to be at the midpoint of their ranges, and after calibration. The figure shows the 'uncalibrated' model responds reasonably to the inputs; it generally captures the timing of flows and emulates the low flow segments well but is overly responsive to large precipitation events, generating spurious spikes in flows. Calibration, either manual by expert knowledge or automatic as done here via optimization, can fix the discrepancies and fit the model output to observations.

So, a fundamental difference between the two approaches is now clearer: using a process-based model is about directly using a wealth of expert knowledge available in a scientific field while using DL is about learning everything from scratch directly from data. This difference is manifest in the number of parameters that need to be tuned to achieve a reasonable performance. Notably, the LSTM model achieved a better performance in emulating observations after calibration, as evident in a comparison of **Figures 10a and b**. However, in any modelling exercise, one needs to ensure the model gives the right answer for the right reasons (Kirchner, 2006). That is why proper model evaluation in out-of-sample prediction is critically important, as discussed in the next section.

7.4. Model validation: Standard versus true out-of-sample prediction

In general, validation and verification of mathematical models are very challenging in some scientific disciplines, if possible at all (Oreskes et al., 1994). The standard practice, however, is to test the performance of the model under investigation in terms of reproducing some historical record not seen during model calibration (Klemeš, 1986a), a process called 'out-of-sample prediction' in this paper. **Figure 9b** shows the results of such practice in the validation period set in **Figure 8**. In this case, both LSTM (standard) and HBV models do reasonably well from a hydrologic point of view, with LSTM outperforming

HBV across all replicates. In addition, and as expected, both models produced slightly lower NSE values in validation compared to those in calibration.

The above so-called ‘model validation’ is inherently partial (Oreskes et al., 1994). While the performance of LSTM appears to be better than that of HBV in a ‘relative’ sense, one needs to take extra care before making such a conclusion. As argued by Klemeš (1986a) more than three decades ago, a strong assumption in this type of validation is that the conditions under which the model will be used will be similar to the conditions under which the model has been developed and calibrated. It is now well-recognized that such an assumption may not hold, as many natural systems are essentially non-stationary (Milly et al., 2008; Razavi et al., 2015). Despite such recognition, this standard model validation practice has arguably remained unchanged (Beven, 2018).

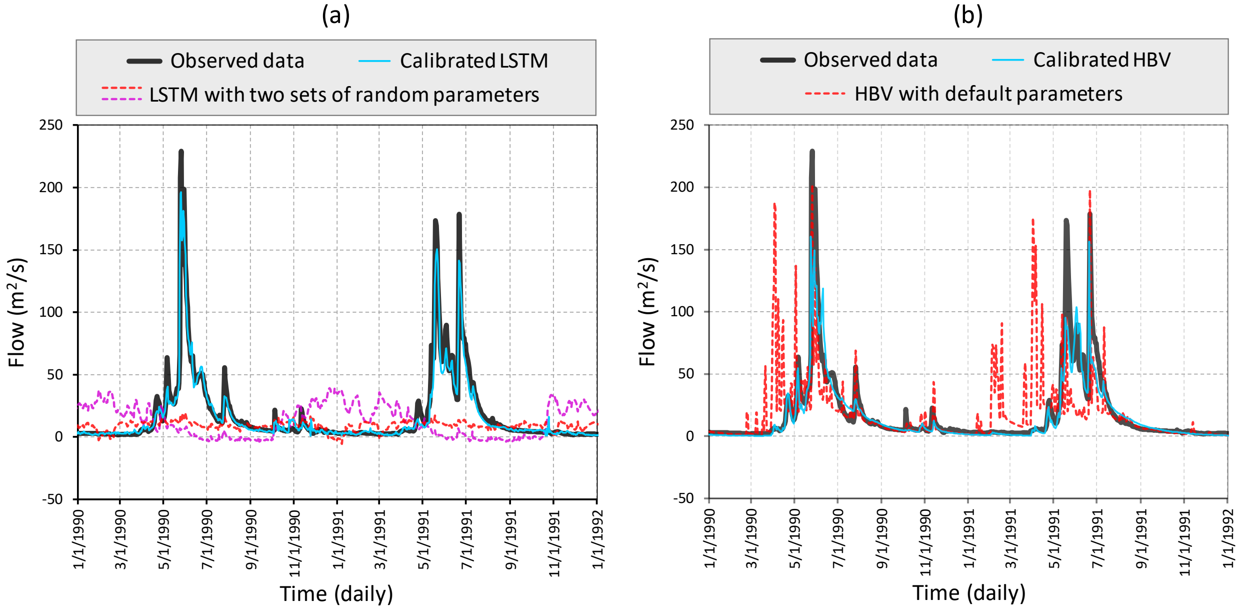

Figure 10. What does a model learn via calibration? Performance samples of (a) LSTM and (b) HBV before and after calibration for a select two-year period.

Here, I took a sensitivity analysis approach via a *what-if scenario* question to test and compare the performance of both models in a ‘true’ out-of-sample prediction, basically under conditions that have not truly been seen in the process of model development and calibration. The question is how the system would behave if the average temperature warmed by 2 °C while everything else remained the same. To assess this scenario, both calibrated models were fed a new temperature time series obtained by adding 2 °C to all daily temperature values of **Figure 8d**. These new ‘synthetic’ inputs roughly provide a picture of what might happen in this watershed under global warming. The modelling results under such scenarios are typically used to inform policy making for climate change adaptation.

Now let us use the two different models to evaluate the possible changes in the watershed behavior in response to a 2 °C warming. Here, instead of looking at individual simulated time series, the possible change in the average seasonality of flows is of interest. First, look at **Figure 11a** to check the consistency of simulated flows for the historical period. Both models generally follow the observed seasonality, but the range provided by the LSTM model is generally narrower and better encapsulates observations in both low and high flows.

Under the new conditions, however, the two models show the two distinct behaviors shown in **Figure 11b**. According to LSTM, peak summer flows would decline by about 25% on average and the time of the peak would shift backward by about a week, from the beginning of June to a time in the fourth week of May. According to HBV-SASK, however, the changes would be more pronounced. The peak flows would decline by about 35% on average and the flows might show two modes: the higher one at the beginning of May and the other at the beginning of June, at about the same time as the peak in the historical observations. Are such differences not sufficiently large so as to make the user skeptical about the modelling process?

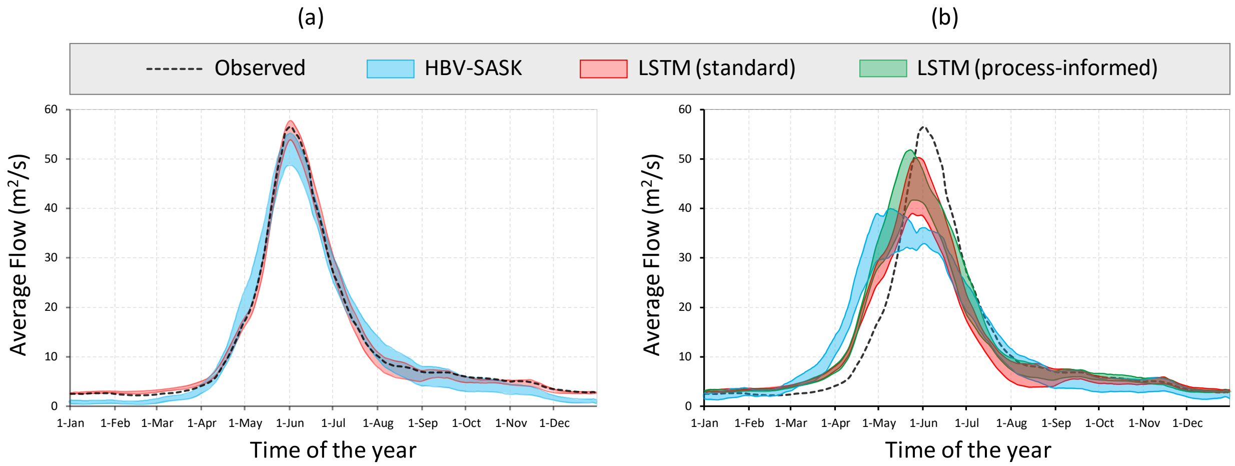

Figure 11. Long-term average daily flows throughout the year under (a) historical and (b) hypothetical conditions. The envelopes represent the daily ranges of flows obtained by the five replicates of each model. The curves were smoothed by a 20-year moving average filter.

7.5. Injecting some physics into ML

At this point, one may wonder about the possibility of ensuring that DL results be physically consistent, particularly under new conditions. Let us give it a try by recasting the modelling problem based on some understanding of the governing physics in hydrology. For example, physics tells us that the freezing point of water is around 0 °C and, therefore, this threshold could be used as an approximation to differentiate rainfall from snowfall on a daily basis, i.e., if the temperature on a day is above/below 0 °C, the precipitation on that day, if any, is considered to be rainfall/snowfall (see **Figures 8b and c**). This differentiation is actually a part of process parameterization in HBV, similar to many other hydrologic models, via a parameter called 'temperature threshold' (TT) for melting/freezing and separating rain and snow, with a feasible range from −4 to +4 °C (see Razavi et al., 2019 for details). The warming of a watershed would naturally change the rainfall to snowfall ratio, and so integrating this domain knowledge with the LSTM model makes sense.

Perhaps the most straightforward way of introducing the TT concept to LSTM is via pre-processing of the inputs. Therefore, a new LSTM model was developed and calibrated, called ‘process-informed LSTM’ in this paper, with three inputs: rainfall, snowfall, and temperature as shown in **Figures 8b, c, and d**. Similar to the original, the new LSTM model has one hidden layer with five neurons, resulting in 186 calibration parameters. The procedure for the calibration and validation of the process-informed LSTM was the same as for the ‘standard LSTM’, already explained in **Sections 7.2** and **7.4**. **Figure 9b** compares the performance of the process-informed LSTM with HBV and the standard LSTM. The figure shows the two LSTM models perform comparably well. Process-informed LSTM results in a slightly lower average NSE in validation but, with only five replicates, this small difference should be interpreted with caution.

Figure 11b demonstrates the performance of the process-informed LSTM model in the ‘true’ out-of-sample prediction. According to this model, the summer peak flows would decline by 20% on average and the time of peak would appear about two weeks earlier than in the historical record, in the third week of May. The process-informed LSTM model generated rising and falling limbs that are more consistent with those of HBV-SASK. Overall, however, the results of HBV-SASK under the new conditions are still quite different.

7.6. So, what model should we trust: the ML or process-based model?

Now the question is which one of the three models produced the most credible picture of possible watershed behavior under the new conditions. In practice, this question is very difficult to answer, if possible at all. In general, the prediction of such changes can be debated and might vary from one study to another, depending on the models and data used and disciplinary views. Perhaps, a definite answer would need to wait until the future has come and shown such possible changes. And, from a bigger-picture point of view, models of natural systems cannot be verified or validated in true out-of-sample prediction, because those systems are never closed and not everything can be represented in a model, as argued by Oreskes et al. (1994) nearly three decades ago.

But, as scientists, we have our own perceptions and intuitions. These might be biased but still useful to provide a ground for building confidence in the credibility of a model. In the context of the case study given, previous research on the Canadian Rocky Mountains has indicated that warming *alone* will result in a considerable reduction in flows and earlier peaks in watersheds similar to the Oldman River watershed. A synthesis of research efforts under the Changing Cold Regions Network (CCRN; DeBeer et al., 2021) on the cold interior of western Canada indicates a shift in timing of the spring hydrograph rise and peak flows of nearly two weeks earlier by mid-21st century, and as much as one month by the late 21st-century. These projections, which themselves are based on rigorous atmosphere-landsurface modelling, are consistent with the modelling results presented in the previous sections but cannot pinpoint the most accurate model.

What is worrisome is the large divergence in behavior of models in response to expected, but yet to be seen perturbations, whereas those models produce comparable results in standard out-of-sample prediction. Broadly speaking, one might say any known consistency of a model with the known underlying physics can improve model’s explainability and interpretability, thereby helping us better explain the model behavior in response to such perturbations. Explainability and interpretability are fundamental assets in building trust in a model, and of course, physically-based models are advantaged in that respect. I would say mistrust in some data-driven modelling paradigms such as ANNs is a long-standing issue in part of our research community and stakeholders. I have heavily struggled with this issue as a researcher who started his research career with ANNs almost 20 years ago and has also had the privilege to work extensively with process-hydrologists and physically-based modellers. I believe with the current momentum and excitement, an opportunity is arising to bring the two world views together and promote the dialogue between the champions of process-based modelling and those of machine learning, as already discussed by many authors (e.g., Reichstein et al., 2019). Doing so, however, requires an in-depth understanding and appreciation of the value of domain knowledge, as discussed in the next section.

8. Discussion

8.1. What is the typically ignored value of domain knowledge in DL?

True out-of-sample prediction is nothing but 'extrapolation' beyond the observed data and behaviors used in model development and calibration. Extrapolation is a reality that many predictive models nowadays must face because of 'non-stationarity' in climate and the environment (Milly et al., 2008; Razavi et al., 2015). Any purely regression-type model, including those arising from DL, would be disadvantaged in extrapolation as, by definition, extrapolating would require working in parts of the problem space for which they have not received any information. Conversely, mechanistic models may be salvaged in extrapolation by the domain knowledge encoded within them.

But what does domain knowledge offer when it comes to extrapolation? The answer is the set of principles modulated via conservation laws (e.g., mass, energy, and momentum) and process parametrizations, which represent our perceptions of how two or more variables might be related (Gupta et al., 2012). Such principles have been developed and evolved over time based on extensive observation and research by scientists and practitioners. The *limits of validity* of such principles are typically known. In the following, the importance of taking advantage of those principles in modelling and prediction is discussed with respect to three aspects: conservation laws, monotonicity and rates, and feedback mechanisms.

Conservation laws: In physics, a conservation law states that a specific measurable property does not change within an isolated system with time. Such a law is usually expressed as a 'continuity equation'; that is, a differential equation equates the rate of change in storage within a control volume with the difference between what comes in and what goes out of the control volume. In land surface modelling, for example, conservation laws are built into mechanistic models to ensure water and energy balance is preserved in simulations over time. ML models, however, do not automatically account for such laws and, as a result, water or energy can be *falsely* introduced or lost in the course of simulation.

Monotonicity and rates: The knowledge base includes the general characteristics of some causal relationships between various physical variables. For example, we know from basic thermodynamics that the relationship between melt rate and available heat is *monotonic*; that is, more heat causes a higher melt rate. Furthermore, we have some rough estimate of the feasible range of the *rate* of change in one with respect to the other. Similarly, from basic hydrology we know the causal relationships governing the way a hillslope stores and releases water are generally such that a positive correlation exists between water available in the soil and its contribution to flows; more water means more flows due to gravitational forces.

Mechanistic models directly account for such knowledge on casual relationships. This knowledge is encoded in process parametrizations typically in the form of deterministic, monotonic functions, or rarely in hysteretic forms, with a limited number of parameters to be calibrated to the specific case study in hand (Gharari and Razavi, 2018). However, in the case of hyper-flexible models such as ANNs, such functions need to be entirely derived from data, all from scratch, and ignoring the knowledge base related to those monotonic relationships. Therefore, extrapolation runs the risk that such relationships become non-monotonic and/or have unrealistic rates, producing erroneous behaviors. This risk is exacerbated by the fact that identifying and diagnosing such errors are very difficult, if possible at all.

Feedback mechanisms: A real-world physical system is a combination of variables that interact over time, typically via a range of feedback mechanisms. Such feedback mechanisms control the internal dynamics of the system and are key to its evolution over time. For example, consider a coupled water-vegetation system in which precipitation, available soil moisture, and plant biomass interact in complex time-dependent ways, even at times creating positive feedbacks that destabilize the system's behavior (Rodriguez-Iturbe et al., 1991; Scheffer et al., 2001). The knowledge base available about these feedback mechanisms is often built into mechanistic models, using differential equations (ordinary or partial) to describe the system dynamics. The representation of such dynamics in the making of models is important, particularly for long-term predictions and over long time scales.

DL models are often unable to account explicitly for such long-term dynamics. If a particular dynamical behavior is present in training data, then DL can capture that behavior in its mapping from input onto output. DL however has no explicit mechanism to represent that dynamic under perturbed conditions beyond what has been recorded in the training data.

Is mechanistic modelling immune to issues with extrapolation? Certainty not. While a discussion on the limitations and prospects of mechanistic modelling is beyond this paper, one solution to improve extrapolability of mechanistic modelling over time that is also relevant to ML is 'space-for-time substitution'. This strategy is to investigate multiple or many sites simultaneously, instead of one, to infer a temporal trend for a site based on information from other sites that have different properties and/or experienced different conditions, assuming spatial and temporal variations are equivalent. For example, refer to Pickett (1989) and Blois et al. (2013) in the context of ecology and to Singh et al. (2011) in the context of hydrology. In the era of big data, ML can benefit significantly, explicitly or implicitly, from such strategies when spatio-temporal data across large domains are available. For example, Kratzert et al. (2019), Feng et al. (2020), and Ma et al. (2021) utilize the CAMELS dataset, which includes catchment attributes and hydrometeorological data across many different sites (Newman et al., 2014; Addor et al., 2017), to improve the performance of DL in hydrological modelling applications.

The bottom line is that mechanistic models are generally expected to be less prone to generating spurious behaviors in true out-of-sample prediction. Therefore, many domain experts may be inclined to trust physically based models as their behavior is constrained by physical laws that are perceived as unchanging with time. The points made in this section will become clearer in the next section, where the essential differences between DL and mechanistic modelling are discussed.

8.2. Why is DL essentially different from process-based modelling?

In the author's view, the first principles of ANNs are rooted in *connectionism, hyper-flexibility,* and *vigorous optimization*. These characteristics are fundamentally different from the guiding principles of developing and calibrating mechanistic models, as described in the following:

- *Connectionism* is an approach that orchestrates a set of simple algebraic operations in a massively parallel manner to create a model that is able to carry out complicated tasks. Following this approach, ANNs represent the response of a system under consideration to an input by summing the collective efforts of many neurons, whose roles cannot be easily attributed to individual processes involved in that system. This is unlike mechanistic modelling where each part of a model is designed to be responsible for a specific process.

- *Hyper-flexibility* is a characteristic of a model with excessive degrees of freedom, which can literally fit any dataset, and is not constrained by the many assumptions held by typical statistical models. ANNs are known to be hyper-flexible. Mechanistic models, however, have limited degrees of freedom depending on the knowledge base available about the processes being modelled. Ideally, mechanistic models tend to have just as many degrees of freedom as can be supported and constrained by available knowledge and data.

- *Vigorous optimization* here refers to the practice of manipulating model parameters at any cost to maximize the goodness-of-fit to calibration data. The training of ANNs is all about minimizing an error function; that is, among two competing ANNs, the one producing smaller errors in calibration and validation is the winner. Optimization is also often an essential part of mechanistic modelling to calibrate model parameters. However, in mechanistic modelling, minimizing the errors is not the goal but a means to improve the realism of the model. In other words, unlike ANNs, physical feasibility of a parameter, its identifiability, and equifinality are key considerations in mechanistic modelling.

The recognition of these fundamental differences is critically important when one aims to choose the right modelling paradigm for a purpose, compare the two paradigms in a case study, or attempt to bridge the two paradigms, possibly for improved modelling performance. The following section outlines the status quo for bridging the two paradigms and some emerging trends.

8.3. How can we bridge DL and process-based modelling?

The history of research on reconciling and bridging ANNs with mechanistic modelling dates back to the early 2000s or perhaps earlier. These efforts have generally had the objective of simultaneously leveraging the strengths of the two modelling paradigms to further our knowledge and predictive ability. Abrahart et al. (2012) reviewed such research in the context of hydrology and refer to it as 'hybridization'. They introduced three possible approaches for this purpose, which herein are referred to as 'surrogate modelling', 'one-way coupling', and 'modular coupling'. Seven years later, Reichstein et al. (2019) in an influential article in Nature re-introduced and proposed the notion of 'hybrid modelling' and the above three approaches as the next steps in earth science. In the following, these three approaches are explained, and then more modern existing approaches arising from research fields beyond earth and environmental sciences are discussed.

Surrogate modelling, alternatively called metamodelling or model emulation, refers to the process of developing and applying a simpler, cheap-to-run model in lieu of a more complex, computationally intensive model (Razavi et al., 2012a). In this process, a data-driven surrogate, such as an ANN, is trained on samples of a limited number of original model runs to approximate the model response surface. The developed surrogate model can then be used in different frameworks in conjunction with the original model, as reviewed in Razavi et al. (2012a), in multi-query applications such as optimization and uncertainty quantification. Example applications of ANNs as surrogates of mechanistic models include Johnson and Rogers (2000), Broad et al. (2005), Behzadian et al. (2009), and Vali et al. (2020).

One-way coupling refers to the process combining a mechanistic model with an ML model such that the output of the former feeds into the latter as input. A general rationale for such a combination is that a mechanistic model may not be able to fully explain the observed data and, therefore, an ML model could be of help in extracting any information left in the residuals of the mechanistic model. For example, consider a case where a mechanistic hydrologic model is used for streamflow forecasting and, as expected, some errors in model outputs are present. An ANN can be used to model such errors over a historical period to provide some predictive ability on the errors for a time step into the future. Then, running these two models in sequence may provide higher forecasting skills. Example applications of such one-way coupling include Shamseldin and O'Connor (2001) and Anctil et al. (2003), and Li et al. (in review).

Modular coupling refers to cases where an ML model is used as a module/sub-model of a larger mechanistic model or *vice versa*. The rationale for this type of coupling may be that a particular model might have proven skills in representing a particular process and is therefore preferred, while other processes are better represented by another model. Hydrologic examples are the work of Chen and Adams (2006) and Corzo et al. (2009), in which ANNs are used as the routing module within a distributed hydrological model. Another example is the work of Chua and Wong (2010) in which an ANN-based hydrologic model takes the output of a kinematic wave model as one of its inputs. And, a recent example is the work of Bennett and Nijssen (2020), in which a DL-based model for the simulation of turbulent heat fluxes is built into a process-based hydrologic model.

Beyond the earth and environmental sciences community, the notion of bridging the knowledge base and ML has a long history (e.g., see the 'knowledge-based artificial neural networks' by Towell and Shavlik (1994)), but it has received significantly more attention recently. Different approaches mostly arising from mathematics and computer science have been proposed under titles such as 'theory-guided data science' (Karpatne et al., 2017), 'informed machine learning' (von Rueden et al., 2019), and 'physics-informed neural networks' (Raissi et al., 2019), to name a few. Providing a full coverage of such approaches is well beyond the scope of this paper, and many of them have been developed for specific application areas with limited relevance to earth and environmental problems. Instead, in the following, I try to be selective and explain three approaches that I found most relevant.

Regularizing ANNs via knowledge-based loss terms. A new regularization function can be developed based on the available knowledge surrounding a given problem and be added to the loss function used in training. For example, any violation of the conservation laws or monotonicity of relationships, as described in **Section 8.1**, can be quantified and penalized during training. Refer to Stewart and Ermon (2017) for an example application of this approach in the context of image processing.

Using mechanistic model runs to augment ANN training data. A mechanistic model can be used to simulate the system under investigation under a range of conditions to generate 'synthetic data' to augment the available training data. This approach may be particularly useful in guiding ANNs in extrapolation beyond conditions seen in the original training data (see the discussion in **Section 8.1**). This approach is based on the assumption that the mechanistic model used is sufficiently accurate—an assumption that needs to be treated with caution. For an example of this approach in the field of systems biology, see Deist et al. (2019).

Integrating differential equations into ANNs. This approach is a very recent and perhaps the most mathematically elaborate in terms of integrating the knowledge base into ANNs, primarily developed by Raissi et al. (2019). It parametrizes the known differential equations describing a system and integrates them into the body of ANNs. The integrated model is then trained to the available data, simultaneously inferring the parameters of the differential equations and network weights. One could view this approach as an extension to the knowledge-based loss terms described above where the new loss term penalizes the network for deviations from those known differential equations. This approach still seems embryonic but perhaps with great potential for scientific breakthroughs.

8.4. What can we learn from prominent DL applications?

As outlined in **Section 1**, DL has already been used across a wide range of disciplines and applications with varying degrees of success. Here, and for context, consider two special and well-known cases of DL applications: playing chess and predicting the stock market. DL has achieved incredible, superhuman-level performance in chess and similar games (Silver et al., 2018), while its performance in stock market prediction has been criticized despite its widespread application (e.g., Pearlstein, 2018). These opposing outcomes may be explained as follows:

- Chess does not possess any properties of ‘complex systems’ (Bar-Yam, 1997), whereas financial systems are essentially *complex*, with a wide range of agents interacting at a wide range of scales, giving rise to emergent behaviors and even black swans. Any AI-based financial services themselves would also be an agent influencing the stock market, even possibly inducing vicious cycles.
- Chess can be viewed as a closed system, as no exogenous factors influence any properties or dynamics of the board and players, whereas stock markets are open systems and, for any analyses, the assumed boundary conditions depend on the analyst’s judgement.
- Chess is a *fully observable* system, as the entire board, pieces, rules, and moves are seen by the players, but stock markets are only *partially observable* and some controlling elements in the market might be hidden to the analysts.
- Chess is *stationary*, as the properties and governing rules of the game remain constant over time, whereas stock markets are *non-stationary* and their long-term dynamics and behaviors may change in unpredictable ways driven by political, social, economic, or natural events.

So what? Earth and environmental systems arguably fall somewhere in between these two specific applications with respect to their four fundamental and inter-related characteristics: such systems are *complex, open*, *partially observable*, and *non-stationary*. Loosely speaking, understanding and predicting earth and environmental systems face similar challenges to those of the stock markets in terms of those four characteristics. However, unlike stock market systems that are conceived to be partially predictable at best (Fama, 1970; Malkiel, 2003), the behaviors of earth and environmental systems are generally believed to be predictable, with limits of predictability that have been improving as more knowledge and data become available.

The comparisons above try to convey two points. First, the revolutionary success of DL in one field of application cannot necessarily be extended directly to another field of application. The context matters, and success depends on the characteristics of the problem at hand. Second, different disciplines may cross-fertilize DL applications and learn from one another. However, cross-fertilization is non-trivial and requires more direct communications between experts in different disciplines about existing methods, common issues, and ways forward.

9. Concluding remarks

Deep learning has perhaps by now served every researcher and practitioner in earth and environmental sciences communities in tasks such as image and language processing, at least through their smart phones. Such astonishing and within-reach technologies have boosted interest in DL, and in AI in general, within these communities, evidenced by the significant growth in the number of their research papers on DL. Many, including the author of this paper, believe the combination of AI with unprecedented data sources and increased computational power will offer exciting new opportunities for expanding our knowledge about various earth and environmental systems. Unsurprisingly, similar to many other innovations, AI and particularly DL techniques are facing different views towards their future; for example, in the hydrology context Nearing et al. (2020) suggest a DL-informed divorce from some of the current hydrological theories while Beven (2020) advocates for the fundamental needs of a knowledge base in DL interpretation.

It is certainly an exciting time for earth and environmental sciences to benefit from DL tools. Shen et al. (2018) picture a bright future but articulate some important technical and cultural challenges to overcome in the years to come, by more targeted educational and organization efforts. We need also to be mindful of any possible risk of hype and over-excitement about the new potential tools. Arguably, still many applications of DL in earth and environmental sciences have primarily focused on off-the-shelf applications of methods largely developed by mathematicians and computer scientists to problems in a new domain with no or limited considerations of the available domain's knowledge base. The immediate risk of such practices is that the popularity of AI tools in earth and environmental sciences would then follow the ups and downs of these tools in the areas from which they originate and the software developed for those purposes. There is also a greater risk, in the author's view, as follows.

Let us flash back to more than three decades ago, when the prominent statistician George Box (1976, p. 797-798) warned about the "mathematistry" trap, "characterized by development of theory for theory's sake, which since it seldom touches down with practice, has a tendency to redefine the problem rather than solve it". He argued that "there is unhappy evidence that mathematistry is not harmless. In such areas as sociology, psychology, education, and even, I sadly say, engineering, investigators who are not themselves statisticians sometimes take mathematistry seriously. Overawed by what they do not understand, they mistakenly distrust their own common sense and adopt inappropriate procedures devised by mathematicians with no scientific experience." This sentiment was then echoed by the prominent hydrologist Vit Klemeš (1986b, p. 177 and p. 185), who said "The danger increases with the proliferation of computerized "hydrologic" models whose cheaply arranged ability to fit data is presented as proof of their soundness and as a justification for using them for user-attractive but hydrologically indefensible extrapolations." He continued, "The danger to hydrology from extrapolations based on mathematistry is that they lead it on the path of bad science."

The point here is that the risk of mathematistry seems to be just as fresh as it must have been back then, particularly when it comes to the application of AI tools in earth and environmental sciences. Due to the very nature of such tools, this risk may even well extend to their original areas of application, party because of their lack of explainability and interpretability (see Rudin, 2019), to a point that such practice has been referred to as a form of modern “alchemy”; see Rahimi and Recht (2017) for the sentiment, LeCun (2017) for a rebuttal, and Hutson (2018) for a summary. This point is not to undermine the benefits of AI technology, particularly for earth and environmental applications. Instead, it calls for improved rigor and better appreciation of the knowledge base available. After all, it has been long known in environmental sciences that complex models can be made to produce virtually any desired behavior given their large degrees of freedom, as articulated by Hornberger and Spear (1981) three decades ago.

Having such risks in mind, the new potential afforded by AI for earth and environmental sciences is great. To realize this potential, we need to reconcile data-driven AI techniques and the theory-driven knowledge base. The knowledge base is at the heart of ‘traditional programming’, which is still a major building block of process-based or mechanistic modelling in earth and environmental sciences. Clearly, the traditional, knowledge-based programming and AI are made up of two fundamentally different world views for problem solving and, therefore, their reconciliation will not be straightforward. This paper tried to address some critical questions in this regard and provide some perspective for this important endeavor, in anticipation of new breakthroughs in earth and environmental sciences in an age of big data and computational power.

Acknowledgements

The author is thankful for funding support from the Natural Sciences and Engineering Research Council of Canada (Discovery Grants) and Integrated Modeling Program for Canada (IMPC) under the framework of Global Water Futures (GWF).

References

Abbott, M.B., (1991). Hydroinformatics: Information Technology and the Aquatic Environment. Avebury Technical. ISBN13 9781856288323.

Abrahart, R. J., Anctil, F., Coulibaly, P., Dawson, C. W., Mount, N. J., See, L. M., ... & Wilby, R. L. (2012). Two decades of anarchy? Emerging themes and outstanding challenges for neural network river forecasting. Progress in Physical Geography, 36(4), 480-513.

Addor, N., Newman, A. J., Mizukami, N., & Clark, M. P. (2017). The CAMELS data set: catchment attributes and meteorology for large-sample studies. Hydrology and Earth System Sciences, 21(10), 5293-5313.

Aizenberg, I., Aizenberg, N. N., & Vandewalle, J. P. L. (2000). Multi-Valued and Universal Binary Neurons: Theory, Learning and Applications. Springer Science & Business Media.

Anctil, F., Perrin, C., & Andréassian, V. (2003). ANN output updating of lumped conceptual rainfall/runoff forecasting models, Journal of the American Water Resources Association, 39(5), 1269-1279.

Ashouri, H., Hsu, K. L., Sorooshian, S., Braithwaite, D. K., Knapp, K. R., Cecil, L. D., ... & Prat, O. P. (2015). PERSIANN-CDR: Daily precipitation climate data record from multisatellite observations for hydrological and climate studies. Bulletin of the American Meteorological Society, 96(1), 69-83.

Badran, F., Thiria, S., & Crepon, M. (1991). Wind ambiguity removal by the use of neural network techniques. Journal of Geophysical Research: Oceans, 96(C11), 20521-20529.

Bankert, R. L. (1994). Cloud classification of AVHRR imagery in maritime regions using a probabilistic neural network. Journal of Applied Meteorology, 33(8), 909-918.

Bar-Yam, Y. (1997). Dynamics of Complex Systems. Perseus Books, USA.

Behzadian, K., Kapelan, Z., Savic, D., & Ardeshir, A. (2009). Stochastic sampling design using a multi-objective genetic algorithm and adaptive neural networks. Environmental Modelling & Software, 24(4), 530-541.

Benediktsson, J. A., Swain, P. H., & Ersoy, O. K. (1990). Neural network approaches versus statistical methods in classification of multisource remote sensing data. IEEE Transactions on Geoscience and Remote Sensing, 28(4).

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2), 157-166.

Benítez, J. M., Castro, J. L., & Requena, I. (1997). Are artificial neural networks black boxes? IEEE Transactions on Neural Networks, 8(5), 1156-1164.

Bennett, A., & Nijssen, B. (2020). Deep learned process parameterizations provide better representations of turbulent heat fluxes in hydrologic models. Earth and Space Science Open Archive ESSOAr.

Bergen, K. J., Johnson, P. A., Maarten, V., & Beroza, G. C. (2019). Machine learning for data-driven discovery in solid Earth geoscience. Science, 363(6433).

Beven, K. (2020). Deep learning, hydrological processes and the uniqueness of place. Hydrological Processes, 34, 3608–3613, https://doi.org/10.1002/hyp.13805.

Beven, K. J. (2018). On hypothesis testing in hydrology: why falsification of models is still a really good idea. Wiley Interdisciplinary Reviews: Water, 5(3), e1278.

Beven, K., & Freer, J. (2001). Equifinality, data assimilation, and uncertainty estimation in mechanistic modelling of complex environmental systems using the GLUE methodology. Journal of Hydrology, 249(1-4), 11-29.

Blois, J. L., Williams, J. W., Fitzpatrick, M. C., Jackson, S. T., & Ferrier, S. (2013). Space can substitute for time in predicting climate-change effects on biodiversity. Proceedings of the National Academy of Sciences, 110(23), 9374-9379.

Bottou, L. (1998). Online learning and stochastic approximations. On-line learning in neural networks, 17(9), 142.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010 (pp. 177-186). Physica-Verlag HD.

Bourlard, H., & Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. Biological Cybernetics, 59(4-5), 291-294.

Box, G. E. (1976). Science and statistics. Journal of the American Statistical Association, 71(356), 791-799.

Boznar, M., Lesjak, M., & Mlakar, P. (1993). A neural network-based method for short-term predictions of ambient SO_2 concentrations in highly polluted industrial areas of complex terrain. Atmospheric Environment. Part B. Urban Atmosphere, 27(2), 221-230.

Broad, D. R., Dandy, G. C., & Maier, H. R. (2005). Water distribution system optimization using metamodels. Journal of Water Resources Planning and Management, 131(3), 172-180.

Broomhead, D. S., & Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. Complex Systems, 2, 321–355.

Cabrera-Mercader, C. R., & Staelin, D. H. (1995). Passive microwave relative humidity retrievals using feedforward neural networks. IEEE Transactions on Geoscience and Remote Sensing, 33(6), 1324-1328.

Castro, J. L., Mantas, C. J., & Benítez, J. M. (2002). Interpretation of artificial neural networks by means of fuzzy rules. IEEE Transactions on Neural Networks, 13(1), 101-116.

Chen, J., & Adams, B. J. (2006). Integration of artificial neural networks with conceptual models in rainfall-runoff modeling. Journal of Hydrology, 318(1-4), 232-249.

Cherkassky, V., & Ma, Y. Q. (2004). Practical selection of SVM parameters and noise estimation for SVM regression. Neural Networks, 17(1), 113–126.

Chua, L. H., & Wong, T. S. (2010). Improving event-based rainfall–runoff modeling using a combined artificial neural network–kinematic wave approach. Journal of Hydrology, 390(1-2), 92-107.

Corzo, G. A., Solomatine, D. P., Hidayat, de Wit, M., Werner, M., Uhlenbrook, S., and Price, R. K. (2009). Combining semi-distributed process-based and data-driven models in flow simulation: a case study of the Meuse river basin. Hydrology and Earth System Sciences, 13, 1619-1634, https://doi.org/10.5194/hess-13-1619-2009.

Dangeti, P. (2017). Statistics for Machine Learning. Packt Publishing Ltd.

de Villiers, J., and Barnard, E. (1993). Backpropagation neural nets with one and two hidden layers. IEEE Transactions on Neural Networks, 4(1), 136-141.

DeBeer, C. M., Wheater, H. S., Pomeroy, J. W., Barr, A. G., Baltzer, J. L., Johnstone, J. F., Turetsky, M. R., Stewart, R. E., Hayashi, M., van der Kamp, G., Marshall, S., Campbell, E., Marsh, P., Carey, S. K., Quinton, W. L., Li, Y., Razavi, S., Berg, A., McDonnell, J. J., Spence, C., Helgason, W. D., Ireson, A. M., Black, T. A., Davison, B., Howard, A., Thériault, J. M., Shook, K., and Pietroniro, A. (2021). Summary and synthesis of Changing Cold Regions Network (CCRN) research in the interior of western Canada – Part 2: Future change in cryosphere, vegetation, and hydrology. Hydrology and Earth System Sciences, 25(4), 1849-1882.

Dechter, R. (1986). Learning while searching in constraint-satisfaction problems. University of California, Computer Science Department, Cognitive Systems Laboratory.

Deist, T. M., Patti, A., Wang, Z., Krane, D., Sorenson, T., & Craft, D. (2019). Simulation-assisted machine learning. Bioinformatics, 35(20), 4072-4080.

Dengiz, B., Alabas-Uslu, C., & Dengiz, O. (2009). A tabu search algorithm for the training of neural networks. Journal of the Operational Research Society, 60(2), 282-291.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Ducournau, A., & Fablet, R. (2016, December). Deep learning for ocean remote sensing: an application of convolutional neural networks for super-resolution on satellite-derived SST data. In 2016 9th IAPR Workshop on Pattern Recognition in Remote Sensing (PRRS) (pp. 1-6). IEEE.

Duerr, O., Sick, , B. and Murina, E., (2020), Probabilistic Deep Learning: With Python, Keras and TensorFlow Probability, Publisher: Simon and Schuster, 296 pages, ISBN: 1617296074, 9781617296079 (available online at: https://livebook.manning.com/book/probabilistic-deep-learning-with-python/chapter-1/72)

Elman, J. L. (1990). Finding structure in time. Cognitive Science, 14(2), 179-211.

Eyring, V., Bony, S., Meehl, G. A., Senior, C. A., Stevens, B., Stouffer, R. J., & Taylor, K. E. (2016). Overview of the Coupled Model Intercomparison Project Phase 6 (CMIP6) experimental design and organization. Geoscientific Model Development, 9(5), 1937-1958.

Fama, E. F. (1970). Efficient capital markets: a review of theory and empirical work. The Journal of Finance, 25(2), 383-417.

Fang, X., & Pomeroy, J. W. (2020). Diagnosis of future changes in hydrology for a Canadian Rockies headwater basin. Hydrology and Earth System Sciences, 24(5), 2731-2754.

Feng, D., Fang, K., & Shen, C. (2020). Enhancing streamflow forecast and extracting insights using long-short term memory networks with data integration at continental scales. Water Resources Research, 56(9), e2019WR026793.

Foresee, F. D., & Hagan, M. T. (1997, June). Gauss-Newton approximation to Bayesian learning. In Proceedings of International Conference on Neural Networks (ICNN'97) (Vol. 3, pp. 1930-1935). IEEE.

Frasconi, P., Gori, M., & Soda, G. (1992). Local feedback multilayered networks. Neural Computation, 4(1), 120-130.

Friedman, J. H. (1991). Multivariate adaptive regression splines. Annals of Statistics, 19(1), 1–67.

Gardner, M. W., & Dorling, S. R. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmospheric Environment, 32(14-15), 2627-2636.

Gelaro, R., McCarty, W., Suárez, M. J., Todling, R., Molod, A., Takacs, L., ... & Wargan, K. (2017). The modern-era retrospective analysis for research and applications, version 2 (MERRA-2). Journal of Climate, 30(14), 5419-5454.

Gharari, S., & Razavi, S. (2018). A review and synthesis of hysteresis in hydrology and hydrological modeling: memory, path-dependency, or missing physics? Journal of Hydrology, 566, 500-519.

Glorot, X., Bordes, A., & Bengio, Y. (2011, June). Deep sparse rectifier neural networks. In Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (pp. 315-323).

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). Deep learning (Vol. 1). Cambridge: MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014). pp. 2672–2680.

Gu, H., Xu, Y. P., Ma, D., Xie, J., Liu, L., & Bai, Z. (2020). A surrogate model for the Variable Infiltration Capacity model using deep learning artificial neural network. Journal of Hydrology, 588, 125019.

Guillaume, J. H., Jakeman, J. D., Marsili-Libelli, S., Asher, M., Brunner, P., Croke, B., ... & Stigter, J. D. (2019). Introductory overview of identifiability analysis: a guide to evaluating whether you have the right type of data for your modeling purpose. Environmental Modelling & Software, 119, 418-432.

Gupta, H. V., Clark, M. P., Vrugt, J. A., Abramowitz, G., & Ye, M. (2012). Towards a comprehensive assessment of model structural adequacy. Water Resources Research, 48(8).

Hagan, M. T., & Menhaj, M. B. (1994). Training feedforward networks with the Marquardt algorithm. IEEE Transactions on Neural Networks, 5(6), 989-993.

Hagan, M. T., Demuth, H. B., & Beale, M. (1996). Neural network design. PWS Publishing Company, Boston, MA.

Hendler, J. (2008). Avoiding another AI winter. IEEE Intelligent Systems, (2), 2-4.

Hinton, G. E., McClelland, J., & Rumelhart, D. (1986). Distributed representations. In D. E. Rumelhart & J. L. McClelland, editors, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, volume 1, pages 77–109. MIT Press, Cambridge.

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural Computation, 18(7), 1527-1554.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780.

Hornberger, G. M., & Spear, R. C. (1981). Approach to the preliminary analysis of environmental systems. J. Environ. Mgmt., 12(1), 7-18.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. Neural Networks, 2(5), 359-366.

Hosny, A., Parmar, C., Quackenbush, J., Schwartz, L. H., & Aerts, H. J. (2018). Artificial intelligence in radiology. Nature Reviews Cancer, 18(8), 500-510.

Hsu, K. L., Gao, X., Sorooshian, S., & Gupta, H. V. (1997). Precipitation estimation from remotely sensed information using artificial neural networks. Journal of Applied Meteorology, 36(9), 1176-1190.

Hsu, K. L., Gupta, H. V., & Sorooshian, S. (1995). Artificial neural network modeling of the rainfall-runoff process. Water Resources Research, 31(10), 2517-2530.

Hutson, M. (2018). Has artificial intelligence become alchemy? Science, 360(6388), 478. DOI: 10.1126/science.360.6388.478

Johansen, T. A. (1997). On Tikhonov regularization, bias and variance in nonlinear system identification. Automatica, 33(3), 441-446.

Johnson, V. M., & Rogers, L. L. (2000). Accuracy of neural network approximators in simulation-optimization. Journal of Water Resources Planning and Management, 126(2), 48-56.

Jordan, M. I. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. In 8th Annual Conference, Cognitive Science Society. MIT Press: Amherst, MA; 531–546.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: trends, perspectives, and prospects. Science, 349(6245), 255-260.

Kang, K. W., Park, C. Y., & Kim, J. H. (1993). Neural network and its application to rainfall-runoff forecasting. Korean Journal of Hydrosciences, 4, 1-9.

Karamouz, M., Razavi, S., & Araghinejad, S. (2008). Long-lead seasonal rainfall forecasting using time-delay recurrent neural networks: a case study. Hydrological Processes: An International Journal, 22(2), 229-241.

Karpatne, A., Atluri, G., Faghmous, J. H., Steinbach, M., Banerjee, A., Ganguly, A., ... & Kumar, V. (2017). Theory-guided data science: a new paradigm for scientific discovery from data. IEEE Transactions on Knowledge and Data Engineering, 29(10), 2318-2331.

Kennedy, M. C., & O'Hagan, A. (2000). Predicting the output from a complex computer code when fast approximations are available. Biometrika, 87(1), 1–13.

Khatami, S., Peel, M. C., Peterson, T. J., & Western, A. W. (2019). Equifinality and flux mapping: A new approach to model evaluation and process representation under uncertainty. Water Resources Research, 55(11), 8922-8941.

Kim, S. S. (1998). Time-delay recurrent neural network for temporal correlations and prediction. Neurocomputing, 20(1-3), 253-263.

Kirchner, J. W. (2006). Getting the right answers for the right reasons: linking measurements, analyses, and models to advance the science of hydrology. Water Resources Research, 42(3).

Klemeš, V. (1986a). Operational testing of hydrological simulation models. Hydrological Sciences Journal, 31(1), 13-24.

Klemeš, V. (1986b). Dilettantism in hydrology: transition or destiny? Water Resources Research, 22(9S), 177S-188S.

Kolakowski, M. (2018). How Algo Trading Is Worsening Stock Market Routs, Investopedia (https://www.investopedia.com/news/how-algo-trading-worsening-stock-market-routs/)

Krasnopolsky, V. M. (2007). Neural network emulations for complex multidimensional geophysical mappings: applications of neural network techniques to atmospheric and oceanic satellite retrievals and numerical modeling. Reviews of Geophysics, 45(3).

Kratzert, F., Klotz, D., Brenner, C., Schulz, K., & Herrnegger, M. (2018). Rainfall–runoff modelling using long short-term memory (LSTM) networks. Hydrology and Earth System Sciences, 22(11), 6005-6022.

Kratzert, F., Klotz, D., Herrnegger, M., Sampson, A. K., Hochreiter, S., & Nearing, G. S. (2019). Toward improved predictions in ungauged basins: Exploiting the power of machine learning. Water Resources Research, 55(12), 11344-11354.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84-90.

Krogh, A., & Hertz, J. (1991). A simple weight decay can improve generalization. Advances in Neural Information Processing Systems, 4, 950-957.

Lawrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. (1997). Face recognition: a convolutional neural-network approach. IEEE Transactions on Neural Networks, 8(1), 98-113.

LeCun, Y. (2017). My take on Ali Rahimi's "Test of Time" award talk at NIPS, (accessed online in December 2020 at: https://www2.isye.gatech.edu/~tzhao80/Yann_Response.pdf)

Lee, J., Kim, R., Koh, Y., & Kang, J. (2019). Global stock market prediction based on stock chart images using deep Q-network. IEEE Access, 7, 167260-167277.

Li, D., Marshall, L., Liang, Z., Sharma, A., and Zhou, Y. (in review), Characterizing distributed hydrological model residual errors using a probabilistic Long Short-Term Memory network, Journal of Hydrology

Lindström, G., Johansson, B., Persson, M., Gardelin, M., & Bergström, S. (1997). Development and test of the distributed HBV-96 hydrological model. Journal of hydrology, 201(1-4), 272-288.

Ma, K., Feng, D., Lawson, K., Tsai, W. P., Liang, C., Huang, X., ... & Shen, C. (2021). Transferring Hydrologic Data Across Continents–Leveraging Data-Rich Regions to Improve Hydrologic Prediction in Data-Sparse Regions. Water Resources Research, 57(5), e2020WR028600.

MacKay, D. J. (1992). A practical Bayesian framework for backpropagation networks. Neural Computation, 4(3), 448-472.

Maier, H. R., & Dandy, G. C. (2000). Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications. Environmental Modelling & Software, 15(1), 101-124.

Maier, H. R., Jain, A., Dandy, G. C., & Sudheer, K. P. (2010). Methods used for the development of neural networks for the prediction of water resource variables in river systems: current status and future directions. Environmental Modelling & Software, 25(8), 891-909.

Malkiel, B. G. (2003). The efficient market hypothesis and its critics. Journal of Economic Perspectives, 17(1), 59-82.

McCann, D. W. (1992). A neural network short-term forecast of significant thunderstorms. Weather and Forecasting, 7(3), 525-534.

Milly, P. C. D., Betancourt, J., Falkenmark, M., Hirsch, R. M., Kundzewicz, Z. W., Lettenmaier, D. P., & Stouffer, R. J. (2008). Stationarity is dead: whither water management? Science, 319, 573-574.

Minns, A. W., & Hall, M. J. (1996). Artificial neural networks as rainfall-runoff models. Hydrological Sciences Journal, 41(3), 399-417.

Minsky, M., & Papert, S. A. (1969). Perceptrons: An Introduction to Computational Geometry. MIT Press.

Nash, J. E., & Sutcliffe, J. V. (1970). River flow forecasting through conceptual models part I—A discussion of principles. Journal of Hydrology, 10(3), 282-290.

Navone, H. D., & Ceccatto, H. A. (1994). Predicting Indian monsoon rainfall: a neural network approach. Climate Dynamics, 10(6-7), 305-312.

Nearing, G. S., Kratzert, F., Sampson, A. K., Pelissier, C. S., Klotz, D., Frame, J. M., ... & Gupta, H. V. (2020). What role does hydrological science play in the age of machine learning?. Water Resources Research, e2020WR028091.

Newman, A. Sampson, K., Clark, M. P., Bock, A. Viger, R. J., Blodgett, D., (2014). A large-sample watershed-scale hydrometeorological dataset for the contiguous USA. Boulder, CO: UCAR/NCAR. https://dx.doi.org/10.5065/D6MW2F4D

Oreskes, N., Shrader-Frechette, K., & Belitz, K. (1994). Verification, validation, and confirmation of numerical models in the earth sciences. Science, 263(5147), 641-646.

Pan, B., Hsu, K., AghaKouchak, A., & Sorooshian, S. (2019). Improving precipitation estimation using convolutional neural network. Water Resources Research, 55(3), 2301-2321.

Panchal, J. H., Fuge, M., Liu, Y., Missoum, S., & Tucker, C. (2019). Machine learning for engineering design. Journal of Mechanical Design, 141(11).

Pearlstein, S. (2018). The robots-vs.-robots trading that has hijacked the stock market, The Washington Post, https://www.washingtonpost.com/news/wonk/wp/2018/02/07/the-robots-v-robots-trading-that-has-hijacked-the-stock-market/

Pickett, S. T. (1989). Space-for-time substitution as an alternative to long-term studies. In Long-term studies in ecology (pp. 110-135). Springer, New York, NY.

Prechelt, L. (1998). Early stopping-but when? In G. Montavon, G. B. Orr, & K.-R. Müller, editors, Neural Networks: Tricks of the Trade, pages 55-69. Springer, Berlin, Heidelberg.

Rahimi, A., and & Recht, B., (2017)., Back when we were kids, Test-of-time award presentation, Conference on Neural Information Processing Systems, 2017, Vancouver, Canada (Video online accessed in December, 2020, at from: https://youtu.be/Qi1Yry33TQE).

Raina, R., Madhavan, A., & Ng, A. Y. (2009, June). Large-scale deep unsupervised learning using graphics processors. In Proceedings of the 26th Annual International Conference on Machine Learning (pp. 873-880).

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). 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.

Rajbhandari, S., Rasley, J., Ruwase, O., & He, Y. (2019). ZeRO: memory optimization towards training a trillion parameter models. arXiv preprint arXiv:1910.02054.

Rakitianskaia, A., & Engelbrecht, A. P. (2009). Training neural networks with PSO in dynamic environments. In 2009 IEEE Congress on Evolutionary Computation (pp. 667-673).

Razavi, S., & Araghinejad, S. (2009). Reservoir inflow modeling using temporal neural networks with forgetting factor approach. Water Resources Management, 23(1), 39-55.

Razavi, S., & Karamouz, M. (2007). Adaptive neural networks for flood routing in river systems. Water international, 32(3), 360-375.

Razavi, S., & Tolson, B. A. (2011). A new formulation for feedforward neural networks. IEEE Transactions on Neural Networks, 22(10), 1588-1598.

Razavi, S., Elshorbagy, A., Wheater, H., & Sauchyn, D. (2015). Toward understanding nonstationarity in climate and hydrology through tree ring proxy records. Water Resources Research, 51(3), 1813-1830.

Razavi, S., Jakeman, A., Saltelli, A., Prieur, C., Iooss, B., Borgonovo, E., ... & Tarantola, S. (2021). The fFuture of sSensitivity aAnalysis: aAn eEssential dDiscipline for sSystems mModeling and pPolicy sSupport. Environmental Modelling & Software, 104954.

Razavi, S., Sheikholeslami, R., Gupta, H. V., & Haghnegahdar, A. (2019). VARS-TOOL: A toolbox for comprehensive, efficient, and robust sensitivity and uncertainty analysis. Environmental Modelling & Software, 112, 95-107.

Razavi, S., Tolson, B. A., & Burn, D. H. (2012a). Review of surrogate modeling in water resources. Water Resources Research, 48(7).

Razavi, S., Tolson, B. A., & Burn, D. H. (2012b). Numerical assessment of metamodelling strategies in computationally intensive optimization. Environmental Modelling & Software, 34, 67-86.

Reed, R. (1993). Pruning algorithms—A survey. IEEE Transactions on Neural Networks, 4(5), 740–747.

Reichstein, M., Camps-Valls, G., Stevens, B., Jung, M., Denzler, J., & Carvalhais, N. (2019). Deep learning and process understanding for data-driven Earth system science. Nature, 566(7743), 195-204.

Rodriguez-Iturbe, I., Entekhabi, D., Lee, J. S., & Bras, R. L. (1991). Nonlinear dynamics of soil moisture at climate scales: 2. Chaotic analysis. Water Resources Research, 27(8), 1907-1915.

Rosenblatt, F. (1957). The perceptron, a perceiving and recognizing automaton (Project PARA), Cornell Aeronautical Laboratory Report No. 85-460-1, Buffalo, New York.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence, 1(5), 206-215.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533-536.

Samek, W., and Müller, K.-R. (2019). Towards Explainable Artificial Intelligence. In Explainable AI: Interpreting, Explaining and Visualizing Deep Learning (pp. 5-22). doi:10.1007/978-3-030-28954-6_1.

Scheffer, M., Carpenter, S., Foley, J. A., Folke, C., & Walker, B. (2001). Catastrophic shifts in ecosystems. Nature, 413(6856), 591-596.

Schmidhuber, J. (2015a). "Deep Learning". Scholarpedia. 10(11), 32832. Bibcode:2015SchpJ..1032832S. doi:10.4249/scholarpedia.32832.

Schmidhuber, J. (2015b). Deep learning in neural networks: an overview. Neural Networks, 61, 85-117.

See, L. M., Jain, A., Dawson, C.W., and Abrahart, R.J. (2008). Visualisation of Hidden Neuron Behaviour in a Neural Network Rainfall-Runoff Model. In: Practical Hydroinformatics: Computational Intelligence and Technological Developments in Water Applications (Abrahart, See, Solomatine, eds.). Springer, pp 87-99.

Shamseldin, A. Y., & O'Connor, K. M. (2001). A non-linear neural network technique for updating of river flow forecasts. Hydrology and Earth System Sciences, 5, 577–598, https://doi.org/10.5194/hess-5-577-2001.

Shen, C. (2018). A transdisciplinary review of deep learning research and its relevance for water resources scientists. Water Resources Research, 54(11), 8558-8593.

Shi, X., Chen, Z., Wang, H., Yeung, D. Y., Wong, W. K., & Woo, W. C. (2015). Convolutional LSTM network: a machine learning approach for precipitation nowcasting. In Advances in Neural Information Processing Systems (pp. 802-810).

Shi, X., Gao, Z., Lausen, L., Wang, H., Yeung, D. Y., Wong, W. K., & Woo, W. C. (2017). Deep learning for precipitation nowcasting: A benchmark and a new model. arXiv preprint arXiv:1706.03458.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., ... & Lillicrap, T. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science, 362(6419), 1140-1144.

Singh, R., Wagener, T., Werkhoven, K. V., Mann, M. E., & Crane, R. (2011). A trading-space-for-time approach to probabilistic continuous streamflow predictions in a changing climate–accounting for changing watershed behavior. Hydrology and Earth System Sciences, 15(11), 3591-3603.

Sorooshian, S., Hsu, K. L., Gao, X., Gupta, H. V., Imam, B., & Braithwaite, D. (2000). Evaluation of PERSIANN system satellite-based estimates of tropical rainfall. Bulletin of the American Meteorological Society, 81(9), 2035-2046.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), 1929-1958.

Stewart, R., & Ermon, S. (2017, February). Label-free supervision of neural networks with physics and domain knowledge. In 31st AAAI Conference on Artificial Intelligence.

Stogryn, A. P., Butler, C. T., & Bartolac, T. J. (1994). Ocean surface wind retrievals from special sensor microwave imager data with neural networks. Journal of Geophysical Research: Oceans, 99(C1), 981-984.

Tamura, S., & Tateishi, M. (1997). Capabilities of a four-layered feedforward neural network: four layers versus three. IEEE Transactions on Neural Networks, 8(2), 251–255.

Tang, G., Long, D., Behrangi, A., Wang, C., & Hong, Y. (2018). Exploring deep neural networks to retrieve rain and snow in high latitudes using multisensor and reanalysis data. Water Resources Research, 54(10), 8253-8278.

Tao, Y., Gao, X., Hsu, K., Sorooshian, S., & Ihler, A. (2016). A deep neural network modeling framework to reduce bias in satellite precipitation products. Journal of Hydrometeorology, 17(3), 931-945.

Tao, Y., Hsu, K., Ihler, A., Gao, X., & Sorooshian, S. (2018). A two-stage deep neural network framework for precipitation estimation from bispectral satellite information. Journal of Hydrometeorology, 19(2), 393-408.

Teoh, E. J., Tan, K. C., & Xiang, C. (2006). Estimating the number of hidden neurons in a feedforward network using the singular value decomposition. IEEE Transactions on Neural Networks, 17(6), 1623-1629.

Tickle, A. B., Andrews, R., Golea, M., & Diederich, J. (1998). The truth will come to light: directions and challenges in extracting the knowledge embedded within trained artificial neural networks. IEEE Transactions on Neural Networks, 9(6), 1057-1068.

Tikhonov, A. N. and V. Y. Arsenin (1977). Solutions of Ill-posed Problems. Winston, Washington DC, 258 pages.

Tokar, A. S., & Markus, M. (2000). Precipitation-runoff modeling using artificial neural networks and conceptual models. Journal of Hydrologic Engineering, 5(2), 156-161.

Torresen, J. (2018). A review of future and ethical perspectives of robotics and AI. Frontiers in Robotics and AI, 4, 75.

Towell, G. G., & Shavlik, J. W. (1994). Knowledge-based artificial neural networks. Artificial Intelligence, 70(1-2), 119-165.

Vali, M., Zare, M., & Razavi, S. (2020). Automatic clustering-based surrogate-assisted genetic algorithm for groundwater remediation system design. Journal of Hydrology, 125752.

Vapnik, V. (1998) Statistical Learning Theory, Wiley, New York.

von Rueden, L., Mayer, S., Beckh, K., Georgiev, B., Giesselbach, S., Heese, R., ... & Walczak, M. (2019). Informed machine learning—A taxonomy and survey of integrating knowledge into learning systems. arXiv preprint arXiv:1903.12394.

Waibel, A., Hanazawa, T., Hintin, G., Shikano, K., Lang, K. J. (1989). Phoneme recognition using time delay neural networks. IEEE Transactions on Acoustics, Speech, and Signal Processing, 37(3), 328–339.

Wexler, R. (2017). When a Computer Program Keeps You in Jail. The New York Times (https://www.nytimes.com/2017/06/13/opinion/how-computers-are-harming-criminal-justice.html?auth=login-google)

Wilby, R. L., Abrahart, R. J., & Dawson, C. W. (2003). Detection of conceptual model rainfall—runoff processes inside an artificial neural network. Hydrological Sciences Journal, 48(2), 163-181.

Xiang, C., Ding, S. Q., & Lee, T. H. (2005). Geometrical interpretation and architecture selection of MLP. IEEE Transactions on Neural Networks, 16(1), 84-96.

Xu, Y., Wong, K. W., & Leung, C. S. (2006). Generalized RLS approach to the training of neural networks. IEEE Transactions on Neural Networks, 17(1), 19–34.

Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent trends in deep learning based natural language processing. IEEE Computational Intelligence Magazine, 13(3), 55-75.

Yu, X., Cui, T., Sreekanth, J., Mangeon, S., Doble, R., Xin, P., ... & Gilfedder, M. (2020). Deep learning emulators for groundwater contaminant transport modelling. Journal of Hydrology, 590, 125351.