Manish Narwaria

Explainable Machine Learning

The importance of a system-centric perspective

The landscape in the context of several signal processing applications and even education [1] appears to be significantly affected by the emergence of machine learning (ML) and particularly deep learning (DL). The main reason for this is the ability of DL to model complex and unknown relationships between signals and the tasks of interest. In particular, supervised DL algorithms have been fairly successful at recognizing perceptually or semantically useful signal information in different applications (e.g., identifying objects or regions of interest from image/video signals or recognizing spoken words from speech signals, such as in speech recognition). In all of these tasks, the training process uses labeled data to learn a mapping function (typically implicitly) from signals to the desired information (class label or target label). The trained DL model is then expected to correctly recognize/classify relevant information in a given test signal. Therefore, a DL-based framework is, in general, very appealing since the features and characteristics of the required mapping are learned almost exclusively from the data without resorting to explicit model/system development.

Scope

The aforementioned focus on implicit modeling unfortunately also raises the issue of lack of explainability/interpretability of the resultant DL-based mapping or the black box problem. As a result, explainable ML/DL is an active research area [2], [3], [4], [5], where the primary goal is to elaborate how the ML/DL model arrived at a prediction. We, however, note that, despite the efforts, the commentary on the black box problem appears to lack a technical discussion from the viewpoint of 1) its origin and underlying reasons and 2) its practical implications on the design and deployment of ML/DL systems. Accordingly, a reasonable question that can be raised is as follows: Can the traditional system-centric approach (which places an emphasis on explicit system modeling) provide useful insights into the nature of the black box problem and help develop more transparent ML/DL systems?

Context and relevance

The answer to this question is a yes. This can be better understood by differentiating between a system-centric approach and a data-centric paradigm [1]. The former generally aims at explicit modeling of the physical process by relying on a priori information and a more analytical perspective. Some examples are the characterization of noise as high-frequency components, the use of gradient or edge (high-frequency) information for shape analysis in image/video signals, exploiting the correlation between signal samples (say for signal compression), locating the test statistic on a known probability density function (e.g., in hypothesis testing), and so on. As a result, system design philosophy and performance analysis remain largely amenable to scrutiny.

By contrast, the data-centric approach (i.e., ML/DL) typically focuses on implicit system modeling by learning a mapping function from input to desired output. This is particularly aided by the powerful modeling capabilities of DL [6] that offer the flexibility of evolving a suitable mapping function, i.e., determining a set of weights from the training data. However, a direct interpretation of the mapping learned by DL is generally difficult, giving rise to the black box problem. Hence, it is reasonable to ask the stated question in the context of how a system-centric approach can help enable a better understanding of the black box problem in ML/DL and its practical implications. This is expected to be crucial for making meaningful progress toward the development of more transparent and explainable ML systems.

Therefore, the primary purpose of this lecture note is to shed light on the stated aspects of explainable ML. We also attempt to provide some perspectives on how to mitigate it from the viewpoint of ML system design. To achieve these objectives, we rely on a system-centric philosophy to develop our arguments. We limit ourselves to an easy-to-understand, yet meaningful, example of a simple low-pass filter. This, in our opinion, is not only convenient but also makes the lecture note accessible to readers from diverse backgrounds.

Digital Object Identifier 10.1109/MSP.2022.3211368
Date of current version: xxxxxx

Prerequisites

This lecture note assumes familiarity with basic concepts in signals and systems.

Problem statement and solution

Let $g: \mathbb{R}^p \to \mathbb{R}^q$ denote the mapping function from an input $\boldsymbol{x} \in \mathbb{R}^p$ to the actual (desired) output $\boldsymbol{y} \in \mathbb{R}^q$. For instance, consider the application of object detection in images where we wish to recognize which of, say, three objects of interest is present in a 100 × 100 image. In this case, we have $p = 100 \times 100$ and $q = 1$; i.e., the output $g(\boldsymbol{x})$ is either 0 or 1 or 2 corresponding to one class (object) label. Similarly, in the scenario of object localization, we wish to determine the location of an object in the image. We may denote this by a bounding box specified by a set of four coordinate points. Accordingly, $p = 100 \times 100$ and $q = 8$. We note that the mapping function g in both stated applications, and indeed in many others, is typically unknown. This is where DL in particular has gained popularity since it can potentially learn a mapping $\hat{g}$ from a set of labeled data. One then hopes that $\hat{g}$ is an accurate estimator of g from a practical viewpoint.

At this point, it would be instructive to describe briefly the basic working of DL using notation. As mentioned, we denote the input to the DL model as $\boldsymbol{x} \in \mathbb{R}^p$; i.e., $\boldsymbol{x}$ is a vector of length p. Then, the q-dimensional output of the DL model $\hat{\boldsymbol{y}} \in \mathbb{R}^q$ is

$$\hat{\boldsymbol{y}} = \hat{g}(\boldsymbol{x}; \boldsymbol{\theta}). \tag{1}$$

Here, $\boldsymbol{\theta} = [\boldsymbol{W}_1, ..., \boldsymbol{W}_{L+1}]$ denotes the parameter consisting of $L + 1$ weight matrices (for simplicity, we ignore the intercept or bias term). The entries of matrices $\boldsymbol{W}_1, ..., \boldsymbol{W}_{L+1}$ denote the weights of the connections among different neurons present in a DL model/architecture [6]. These neurons are arranged in a DL model through L hidden layers. As an illustration, we show in Figure 1 a DL model with three hidden layers; i.e., $L = 3$. Notice that in this example the first two hidden layers have three neurons, while the third has two neurons. Moreover, let us assume that $p = 2$ and $q = 1$. Hence, the input and output layers have two neurons and one neuron, respectively. The reader can observe from Figure 1 that there are four weight matrices: $\boldsymbol{W}_1, \boldsymbol{W}_2$, and $\boldsymbol{W}_3$ correspond to the three hidden layers, while $\boldsymbol{W}_4$ represents the weight matrix for the output layer. To compute the output $\hat{\boldsymbol{y}}$, the DL model uses $\boldsymbol{W}_1, ..., \boldsymbol{W}_4$ in series. Also, a nonlinearity is introduced at each neuron via the use of the function $f_a : \mathbb{R}^c \to \mathbb{R}^c$. It is the elementwise nonlinear function commonly referred to as an activation function. Thus, the output can be written as

$$\hat{\boldsymbol{y}} = f_a(\boldsymbol{W}_4 f_a(\boldsymbol{W}_3 f_a(\boldsymbol{W}_2 f_a(\boldsymbol{W}_1 x)))). \tag{2}$$

Note that because we have chosen $p = 2$ and $q = 1$ for the example in Figure 1, the dimensions of $\boldsymbol{x}$, $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, $\boldsymbol{W}_3$, $\boldsymbol{W}_4$, and $\hat{\boldsymbol{y}}$ will be 1 × 2, 2 × 3, 3 × 3, 3 × 2, 2 × 1, and 1 × 1, respectively. The goal of training the DL model is [6] to find the parameter $\boldsymbol{\theta} = [\boldsymbol{W}_1, ..., \boldsymbol{W}_{L+1}]$ via training on a set of labeled data such that $\hat{\boldsymbol{y}}$ is close to $\boldsymbol{y}$ (the actual or desired output).

Problem statement

We note that there are two aspects of a DL system: design and validation. The former refers to the choice of DL architecture (e.g., activation function f_a, number of layers L, number of neurons in each layer, and so on) and subsequent optimization [6] to find the parameter $\boldsymbol{\theta}$. The latter refers to application-specific benchmarking of the trained DL model on an independent test set. Thus, both aspects of the DL model depend heavily on data. As already mentioned, we refer to this as a data-centric approach. Consequently, one typically relies only on implicit modeling (i.e., without the need for explicit signal analysis or handcrafted signal features) and the prediction accuracy as surrogates to explicit system analysis. Such a data-centric approach is in contrast to a system-centric approach, which is typically based on explicit modeling and a priori knowledge. Therefore, our problem statement follows naturally and can be stated as follows: What additional and practically useful insights can a system-centric approach reveal that can eventually help in the design of more transparent and explainable ML/DL systems?

Solution

To identify practically meaningful insights about the explainability aspects of a DL system, we rely on the idea of exploiting explicit a priori knowledge, which is fundamental to the system-centric approach. For instance, channel modeling in communication systems can exploit knowledge of a probabilistic model for the channel filter taps (e.g.,

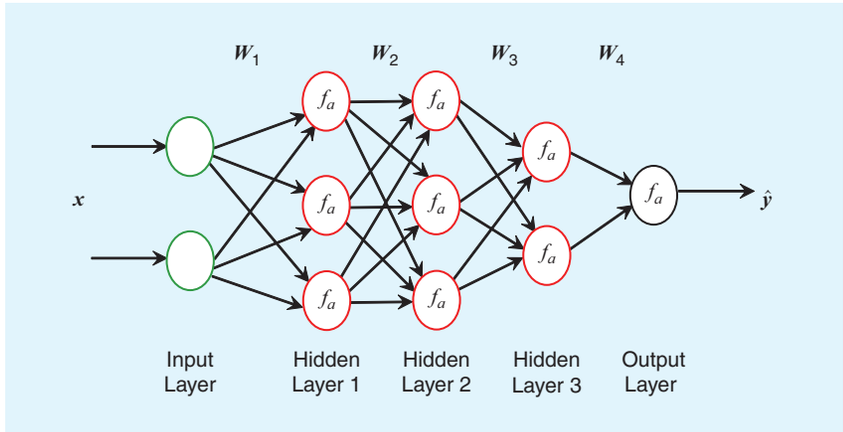

FIGURE 1. A fully connected feedforward network with three hidden layers ($L = 3$). The first two hidden layers have three neurons, while the third has two neurons. As $p = 2$ and $q = 1$ in this example, the input and output layers have two neurons and one neuron, respectively. f_a denotes the activation function, and $\boldsymbol{\theta} = [\boldsymbol{W}_1, \ldots, \boldsymbol{W}_4]$ is the parameter learned via training. The entries of $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_4$ denote the weights of the connections between the neurons. Refer to Figure 2 for an example.

using a Rayleigh distribution). Similarly, characterizing visual signals at different frequencies and orientations (e.g., by using Gabor filters) attempts to explicitly mimic the frequency and orientation selectivity of the human visual system in different applications (such as texture analysis).

Hence, the system-centric approach typically attempts to approximate g via explicit characterization of different subsystems (components) of the physical process/application under consideration. This, in turn, allows a multidimensional analysis of the strengths and weaknesses of each subsystem explicitly from the viewpoint of a certain established knowledge base in the context of the application. Inspired by this philosophy, we pursue the idea of using an explicit system S (with known g) as a reference. We then attempt to model S using DL; i.e., we train a DL network such that the learned function $\hat{g} \approx g$. Because S is explicit by choice, an analysis of the resultant DL-based model of S can reveal additional insights in the context of our problem statement.

Several choices of S are possible. But in this lecture note, we select a simple filtering application where we wish to attenuate perceptually less relevant signal information. This is a common use case in both traditional (signal denoising, smoothing, antialiasing, and so on) and recent application areas (such as spatial audio rendering in augmented and virtual reality, the design of smart cameras for the Internet of Things, and medical imaging, among others). Accordingly, we let S be a low-pass (moving average) filter and use it, as an example, to filter out signal information beyond 2 kHz. Apart from its conceptual simplicity, the said choice of S also enables a fairly straightforward DL-based implementation from the perspective of generating training data and subsequent optimization. Now, from a system-centric perspective, S is conveniently described by the following difference equation:

$$\begin{aligned} \boldsymbol{y}[n] &= \frac{1}{M}\sum_{k=0}^{M-1} \boldsymbol{x}[n-k] \\ &= \sum_{k=0}^{M-1} \boldsymbol{w}[k]\boldsymbol{x}[n-k], \end{aligned} \tag{3}$$

where $k = 0, 1, ..., M-1$. Note that the system S denoted by (3) is explicit and can also be visualized in the frequency domain for a clearer physical interpretation. Setting $M = 2$, we arrive at a simple low-pass filter with filter coefficients $\boldsymbol{w}[0] = \boldsymbol{w}[1] = 0.5$. Therefore, the output is simply the average of the present and past samples in the input $\boldsymbol{x}[n]$, i.e.,

$$\boldsymbol{y}[n] = 0.5\boldsymbol{x}[n] + 0.5\boldsymbol{x}[n-1]. \tag{4}$$

We now attempt to model S using DL-based regression. To that end, we denote the training data as $\{(\mathbf{z}_i, \mathbf{t}_i)\}_{i=1,...,T}$. Because $M = 2$, $\mathbf{z}$ is a $T \times 2$ matrix with $\mathbf{z}_i$ being the ith row. $\mathbf{t}$ denotes a $T \times 1$ vector of the target (desired) values. Given our problem setting, $\mathbf{t}_i$ simply denotes the average of the two numbers in $\mathbf{z}_i$. T represents the size of the training data. The training process seeks to estimate the function $\hat{g}$ such that $\mathbf{t}_i = \hat{g}(\mathbf{z}_i, \boldsymbol{\theta}^*)$, where the parameter $\boldsymbol{\theta}^*$ minimizes a chosen loss function ℓ, i.e.,

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \ell(\mathbf{z}, \mathbf{t}, \boldsymbol{\theta}). \tag{5}$$

We employed the widely used mean square error (MSE) as the loss function, i.e.,

$$\ell(\mathbf{z}, \mathbf{t}, \boldsymbol{\theta}) = \frac{1}{T}\sum_{i=1}^{T} \left\| \mathbf{t}_i - \hat{g}(\mathbf{z}_i, \boldsymbol{\theta}) \right\|^2. \tag{6}$$

Accordingly, we seek a mapping function such that the sum of squared differences between the predicted value $\hat{g}(\mathbf{z}_i, \boldsymbol{\theta})$ and the corresponding target value $\mathbf{t}_i$ is minimized, i.e., $\ell(\mathbf{z}, \mathbf{t}, \boldsymbol{\theta}) \leq \epsilon$. Here, ϵ represents the tolerance or the maximum error allowed during model training. Setting an appropriate value of ϵ is crucial and typically depends on applications as well as the size of training data T. In this lecture note, we are not overly concerned about these aspects and simply choose $\epsilon = 10^{-4}$ and $T = 1{,}000$ for our experiments. In addition, a complete specification of a DL-based model/architecture requires several hyperparameters [6], including the number of hidden layers L, the number of neurons in each hidden layer, the type of activation function f_a, and the loss function ℓ. Hence, these need to be chosen before a DL model can be trained. Once the DL model is trained properly, one expects that it will generalize well to data that did not appear in the training set. In other words, it is hoped that the prediction $\hat{g}(\mathbf{z}_{\text{test}}, \boldsymbol{\theta})$ for any test signal $\mathbf{z}_{\text{test}}$ is close to the unknown target y_{test} (i.e., it is similar to how the trained DL model behaved for the training data). In this lecture note, we used three commonly used activation functions, namely, sigmoid $f_a(x) = 1/(1 + e^{-x})$, rectified linear unit (ReLU) $f_a(x) = \max(0, x)$, and leaky ReLU $f_a(x) = \begin{cases} \alpha x, & x < 0 \\ x & x \geq 0 \end{cases}$. We thus obtained three trained DL networks, namely, $\mathbf{N}_s, \mathbf{N}_r$, and $\mathbf{N}_{lr}$ (f_a = sigmoid, f_a = ReLU, and f_a = leaky ReLU, respectively) with respective functional approximations $\hat{g}_s, \hat{g}_r$, and $\hat{g}_{lr}$. These networks have a single hidden layer ($L = 1$) and are shown in Figure 2. Another network, namely $\mathbf{N}_{lr}^{(3)}$ (with approximation function $\hat{g}_{lr}^{(3)}$), was also trained with $L = 3$ and f_a = leaky ReLU. This network is shown in Figure 3.

Further, we employed ℓ = MSE for training all four DL networks. Note that all of these DL models/networks resulted in $\epsilon \approx 10^{-4}$ on the training dataset. Here, one also needs to ensure mitigation of the issue of overfitting (or memorization), i.e., a DL model performing well on training data but giving a relatively high prediction error on a test set. To that end, a practical and well-accepted solution is to examine the performance of the trained DL model on a test set that is independent of the training data [6]. Accordingly, we cross-validated the performance of the trained DL model on an independent test dataset with 200 data points. We found that all four trained DL models resulted in an error $\approx 10^{-4}$; i.e., the performance was similar to that on the training dataset. Thus, $\mathbf{N}_s, \mathbf{N}_r, \mathbf{N}_{lr}$, and $\mathbf{N}_{lr}^{(3)}$ represent DL-based models of S.

A closer look at the black box problem

As mentioned, explainable ML/DL seeks to get some insights into the black box (the trained ML/DL model)

by elaborating why a trained ML/DL model arrived at a particular prediction. However, there appears to be a lack of discussion on what really is meant by the black box nature of a trained DL model in the first place. Therefore, to understand the issue more closely, it is convenient to first analyze why S as represented by (4) is not a black box. Writing (4) in the frequency domain, we get

$$\begin{aligned} Y(e^{j\Omega}) &= 0.5(1+e^{-j\Omega})X(e^{j\Omega}) \\ &= H(e^{j\Omega})X(e^{j\Omega}), \end{aligned} \tag{7}$$

where we use capital letters to denote the discrete-time Fourier transform of the corresponding signals. The symbol Ω represents the frequency of discrete signals (thus, principally, $\Omega \in [-\pi, \pi]$). Expression (7) tells us that the output $Y(e^{j\Omega})$ of S is simply an elementwise multiplication of the input $X(e^{j\Omega})$ with the system transfer function $H(e^{j\Omega})$. Thus, observing the magnitude of $H(e^{j\Omega})$ shown in Figure 4(a) allows us to understand the explicit nature of S. That is, the system S essentially scales down (reduces) the strength of higher frequency components in the input. This analysis is beneficial from a practical perspective for the following reasons. First, it can provide prior meaningful insights and analysis about system output without actual implementation. For instance, a frequency Ω_0 cannot occur in the output if it does not exist in the input. Second, it allows us to conceptualize and eventually design systems (algorithms) for, say, signal denoising or smoothing, antialiasing, equalizing, and so on. This is enabled by the fact that undesired signals in the input (e.g., noise or perceptually irrelevant components) can be appropriately characterized and then attenuated.

As an example and as shown in Figure 4(a), let us set the cutoff frequency $\Omega_c = \pi/2$ rad/sample; i.e., $0.7 \le |H(e^{j\Omega})| \le 1$ for $0 \le \Omega \le \pi/2$ (or, equivalently, $|H(e^{j\Omega})| < 0.7$ for $\pi/2 < \Omega \le \pi$). Assuming the sampling frequency f_s to be 8 kHz, we observe that S attenuates, from a practical perspective, components beyond 2 kHz. It is, of course, possible to suitably modify the filtering characteristic depending on the application (e.g., by choosing another cutoff frequency). Thus, (7) and Figure 4(a) provide quantifiable insights into the working of S. Such insights also enable practically useful generalizations of S to other use-case scenarios.

With the mentioned aspects of the system-centric approach in mind, it now becomes convenient to examine if the corresponding DL-based model of S is amenable to a similar analysis or not. To that end, we take a closer look at the trained DL network $\mathbf{N}_r$, which is shown in Figure 2, and begin by writing the explicit input–output relationship. Let $\mathbf{z}_{\text{test}} = [x_1 \;\; x_2]$, where $x_1, x_2 \in \mathbb{R}$. Then, using the weight matrices $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ of the trained model $\mathbf{N}_r$, we can write the expression for y_{test} as

$$\begin{aligned} y_{\text{test}} = \max\{0, (0.8283 \max(0, 0.6994\, x_1 \\ + 0.7760\, x_2) - 0.1796 \\ \max(0, 0.4329\, x_1 + 0.8067\, x_2))\}. \end{aligned} \tag{8}$$

Now, by design, $\mathbf{N}_r$ is practically equivalent to S. This, in turn, implies that the corresponding function $\hat{g}_r$ defined by (8) essentially approximates a low-pass filter by using a weighted combination of nonlinear $\max(\cdot)$ functions. However, establishing the low-pass nature of $\mathbf{N}_r$ from an analysis of (8) may be difficult, and this has seri-

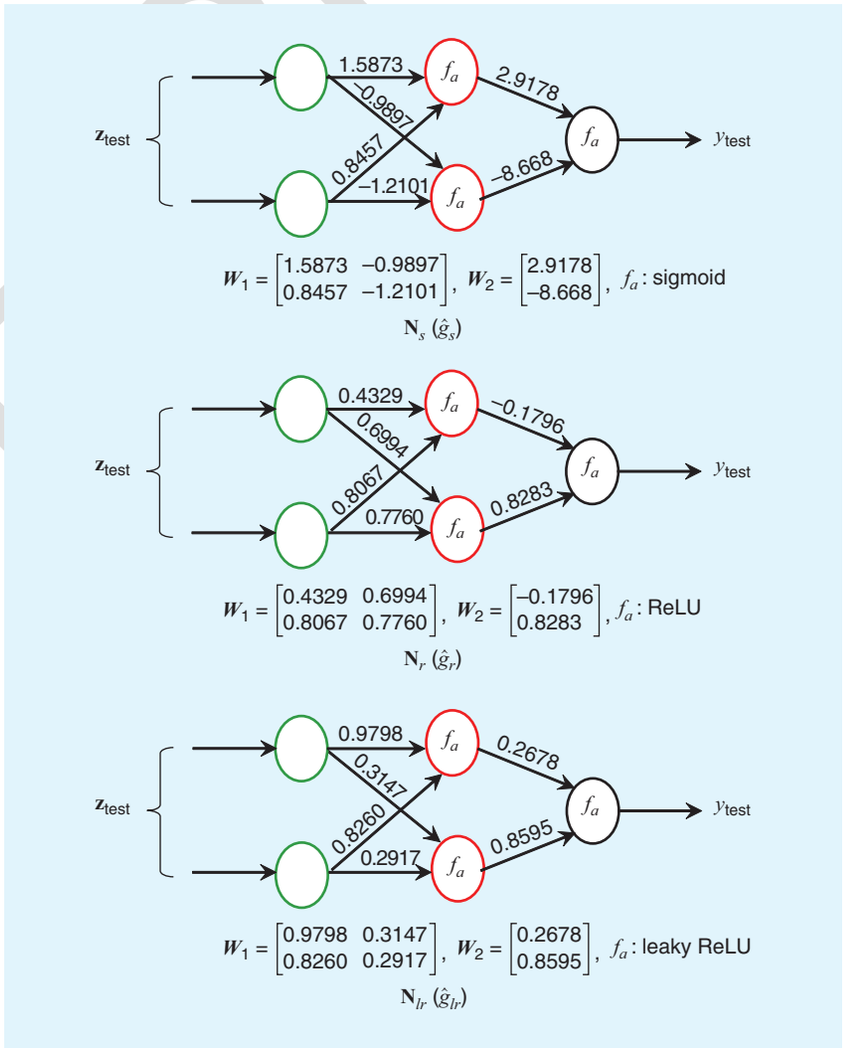

FIGURE 2. The three trained DL networks $\mathbf{N}_s, \mathbf{N}_r$, and $\mathbf{N}_{lr}$, which generate the respective approximation functions $\hat{g}_s, \hat{g}_r$, and $\hat{g}_{lr}$ and, by design, $\hat{g}_s = \hat{g}_r = \hat{g}_{lr} = g$. The input, hidden, and output layers are shown, respectively, in green, red, and black for clarity. Information about trained weights and activation functions is given below each network.

ous implications from a practical perspective. Specifically, it means that we do not know how exactly the trained weights in $\mathbf{N}_r$ (refer to matrices $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$) relate to its low-pass filtering characteristic. This lack of practically meaningful connection between the trained weights and their effect on the input signal essentially constitutes the black box nature of DL. One can similarly analyze the models $\mathbf{N}_s$, $\mathbf{N}_{lr}$, and $\mathbf{N}_{lr}^{(3)}$ and arrive at conclusions similar to that for $\mathbf{N}_r$.

We will now analyze the limitations of the said black box issue from a practical perspective. Before doing that, we note that the presence of unknown and random noise in the training data $\{(\mathbf{z}_i, \mathbf{t}_i)\}_{i=1,\ldots,T}$ and the choice of the loss function (e.g., MSE in our case) will also affect optimization. As a result, the weight matrices $\boldsymbol{W}_1$ to $\boldsymbol{W}_4$ (in Figures 2 and 3) might change. This will, however, not affect the analysis and conclusions made as long as the resultant DL models can approximate S.

Practical implications of the black box problem

Despite its seemingly theoretical underpinnings, a deeper understanding of the black box nature in ML/DL is also important from a practical viewpoint. To elaborate on this, it is convenient to consider another use case where perceptually irrelevant signal information now lies beyond, say, 1.25 kHz (and not 2 kHz). Obviously, S cannot directly serve the purpose in this case. However, from a system-centric perspective, it is fairly straightforward to see that the required filter can be constructed by using $M = 3$ in (3). Thereafter, an analysis similar to that of S in (7) and Figure 4(a) can be carried out. Setting a cutoff frequency $\Omega_c = 5\pi/16$ rad/sample, as visually illustrated in Figure 4(b), will then result in the desired system S'. Thus, while S and S' are two different systems (filters), they are essentially unified via an interpretable and analytical philosophy.

However, the situation is very different in the case of $\mathbf{N}_r$. Specifically, there may not be a general and interpretable procedure of modifying the weights of $\mathbf{N}_r$ such that the resultant model, say $\mathbf{N}_r'$, has a cutoff frequency of 1.25 kHz. The reason is, to reiterate, the lack of meaningful connection between the trained weights and the filtering characteristic of $\mathbf{N}_r$. Thus, the analysis of S afforded by (7) and Figure 4(a) not only offers clear insights into S but also provides a systematic approach to extend the scope to related practical use cases. Likewise, other practical aspects, such as controlling ripples in the passband or controlling gain in the transition band, can be explicitly handled in S. By contrast, DL-based modeling lacks such practically meaningful functionality due to the black box nature.

Another notable practical limitation of the black box problem can show up when the ML/DL needs to be deployed in constrained environments where, say, latency, privacy, or lack of communication bandwidth can be important factors. In this use case, the prediction and update of the DL trained model must take place with limited computing resources, for instance, on a local embedded processing near the sensor or on the edge servers [7]. For that reason, the number of parameters/weights of a DL model (which is one of the measures of model complexity) must typically be

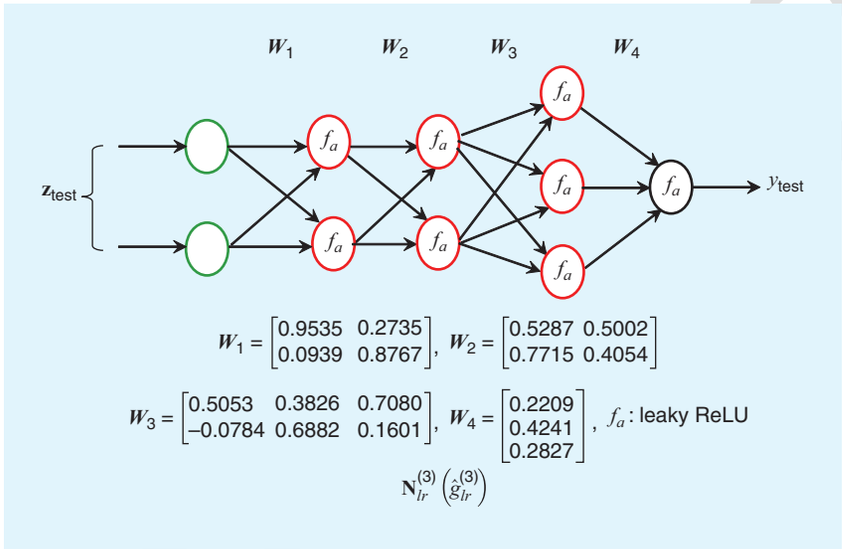

FIGURE 3. Another trained DL network $\mathbf{N}_{lr}^{(3)}$ (with $L = 3$), which generates the approximation function $g_{lr}^{(3)}$ such that $g_{lr}^{(3)} = g$. The input, hidden, and output layers are shown, respectively, in green, red, and black. Information about the trained weights and activation function is also given below the network. For visual clarity, the weights are not shown on the connections between neurons.

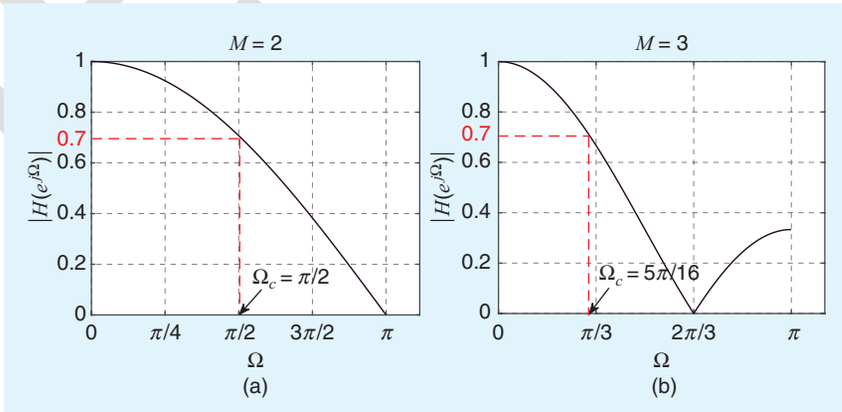

FIGURE 4. Plots of magnitude response in the range $[0, \pi]$ for two values of M. (a) $M = 2$ with the cutoff frequency chosen as $\Omega_c = \pi/2$ (i.e., 2 kHz). (b) $M = 3$ with the cutoff frequency chosen as $\Omega_c = 5\pi/16$ (i.e., 1.25 kHz).

reduced while maintaining practically reasonable prediction performance. (Model complexity also includes other components [7], such as the number of mathematical operations needed, memory requirement, expressive capacity, and so on.)

In this context of on-device/on-edge computation, the black box nature can be a bottleneck in both DL model design and compression (reduction in the number of weights). The reason is that the lack of a clear and quantifiable connection between the trained weights and DL model functionality may prohibit a systematic understanding of the importance of the weights. As a result, making meaningful decisions about DL model reduction may become difficult and prone to trial and error. For instance, aspects such as which weights to quantize more (or less) and the extent of quantization, and/or which connections to prune (i.e., setting some of the weight values to 0), and so on, may remain unclear. On the other hand, the system-centric approach can provide more flexibility toward explicitly analyzing the filtering characteristic of S if $\boldsymbol{w}$ is changed for any reason.

The black box nature will also, in general, prohibit meaningful analysis of the DL model from the viewpoint of its weaknesses (if any). Therefore, in this aspect, one might be constrained to merely analyze the output of DL and see if it matches the desired output or not (an almost exclusive focus on prediction accuracy). This potentially leaves room in terms of direct DL system analysis. In comparison, the system-centric approach allows a more direct system analysis. As an illustration, we may consider the system S'. As can be seen from Figure 4(b), the transfer function of S' shows a sidelobe; i.e., the magnitude increases slightly from 0 at $2\pi/3$ to about 0.35 at frequency π. Obviously, such sidelobes should be as small as possible in the light of the use case of attenuating information beyond 1.25 kHz. Hence, this sidelobe represents a limitation of S' and encourages steps for mitigating the same. A similar functionality in the corresponding DL-based system, say, $\mathbf{N}_r$, might not be possible owing to the black box nature.

Finally, the reader will note from Figure 2 that $\mathbf{N}_s, \mathbf{N}_r$, and $\mathbf{N}_{lr}$ have just six trainable weights. Hence, these DL models are not really complex from the viewpoint of dimensionality of the trained weights, especially in comparison to several well-known DL-based architectures in vision (e.g., a 50-layer ResNet [8] has more than 25 million trained weights). Yet, all of them are essentially black boxes for the purpose at hand. Thus, the black box problem may not necessarily be attributed to high dimensionality of the trained weights alone. Instead, as we have illustrated, it is essentially a practical issue related to a lack of meaningful association between the trained weights and the corresponding functionality of the DL model.

Why is explainability fundamental to DL system design?

We note that the need for explainable ML/DL is largely fueled by high stakes applications [2], like health care, autonomous driving, law enforcement, finance, and so on. Indeed, there is no denying that unexplained mistakes or wrong decisions made by a black box model in such applications can have serious implications. However, it is also equally interesting to note that explainability is, in fact, a fundamental concept for ML/DL system design and not merely a post-design requirement. To appreciate this aspect, it is once again convenient to think from a system-centric perspective and consider our example of S. Observe that S is uniquely characterized by (4) or equivalently via (7). As a consequence, any analysis of S (or even S') from the perspective of its explainability, performance (strengths and/or weakness), implementation issues, and so on can be carried out in an unambiguous fashion. However, this is not the case with DL-based modeling of S as all four models, $\mathbf{N}_s, \mathbf{N}_r, \mathbf{N}_{lr}$, and $\mathbf{N}_{lr}^{(3)}$, accurately mimic the functionality of S. That is, we have $g = \hat{g}_s = \hat{g}_r = \hat{g}_{lr} = \hat{g}_{lr}^{(3)}$. In fact, one can train several other DL models by choosing different L, number of neurons in each layer, f_a, and so on. Consequently, g can be potentially approximated by a large number of DL models. This may seem useful at first sight. However, a closer scrutiny will reveal that such a nonunique DL-based approximation leads to more questions than answers about the unknown function g. In particular, one can raise the following questions:

1) Which DL models' explanation should one rely upon to get accurate insights about the underlying system represented by g and why?
2) Since all the DL models under question approximate g quite well, should they all end up having the same/similar explanations?

In the context of the first question, we may lack a systematic procedure to zero in on one DL model (out of several candidates, such as $\mathbf{N}_s, \mathbf{N}_r, \mathbf{N}_{lr}$, and $\mathbf{N}_{lr}^{(3)}$) and its corresponding explanation as a surrogate to g. The second question also reveals interesting facet of DL modeling. If the answer to it is yes, then again there may be difficulties in establishing the equivalence of the said DL models, which have different architectures (in terms of L, number of neurons in hidden layers, and activation functions). On the other hand, if the answer to the second question is no, then we are potentially looking at a scenario where we have different DL-based explanations for the same physical process (as defined by g in our example). From a practical perspective, this may not be very meaningful. Thus, both questions essentially emphasize why explainability should be an inherent design principle and one of the first in ML/DL system design and not merely a post hoc analysis procedure.

The case for system-centric-philosophy-based explainable ML

We have shown that the philosophy of explicit modeling can be leveraged to understand the black box nature and its practical implications in DL-based systems. It is therefore logical to think that the same system-centric philosophy can also benefit DL-based system design. Indeed, it is possible to exploit a priori domain knowledge and explicit tools for both DL system analysis and design. More specifically, from a DL system

design perspective, there can be several practically useful implications toward

1) preprocessing of the input data to overcome limitations of the DL model itself (e.g., utilizing the idea of Fourier feature mapping to overcome the weakness of the standard multilayer perceptron [9])
2) incorporating aspects of an explicit system into a learning-based framework (e.g., the use of deep unfolding for image superresolution [10] or in communication systems [11])
3) improving certain aspects of the DL model itself, like performance (for instance, the use of the discrete Fourier transform to improve pooling aspects [12] in convolutional neural networks or understanding connections between DL and graph signal processing [13])
4) avoiding training of the DL model from scratch and instead building upon simpler and interpretable aspects of known tools (e.g., the differentiable digital signal processing framework [14] adapts interpretable digital signal processing tools to diverse data via learning for audio applications)
5) visualizing the learning process in DL layers (e.g., using principal component analysis to detect adversarial examples [15]), developing an interactive visual analytics framework in the context of explainable DL ([16], [17]), or creating a design space of explainable systems for medical applications ([18]).

The aforementioned points (not an exhaustive list by any means) emphasize the potential benefits of incorporating a system-centric philosophy in the design and analysis of better performing but more explainable ML/DL systems. Essentially, this strategy of designing system-centric DL models can be a two-step process: 1) developing a base model that might rely more heavily on a priori domain knowledge and 2) refining the resultant base model via learning from application-specific data. This should lead to DL systems that may not only perform better (both in terms of generalization and accuracy) but are also transparent and amenable enough for a practically meaningful scrutiny.

What we have learned

The black box nature of ML/DL represents a fundamental problem. In this context, a system-centric perspective can lead to a more in-depth understanding of this issue from the viewpoint of its origin and practical implications. It also helps to appreciate why explainability represents a fundamental aspect of ML/DL system design. Such understanding may not only improve learning outcomes but can also provide meaningful insights toward improved practical deployment of ML/DL systems. This deployment can span a broad canvas, ranging from design considerations (including enhanced explainability, better generalization, proper model initialization and training, and so on) to hardware implementations (on-device and under constrained environments).

Acknowledgment

The author acknowledges funding from the Science and Engineering Research Board, Department of Science and Technology, Government of India, vide Grant SRG/2020/000849. Manish Narwaria is the corresponding author.

Author

Manish Narwaria (narwaria@iitj.ac.in) received his Ph.D. degree in computer engineering from Nanyang Technological University, Singapore, in 2012. Currently, he is working as an assistant professor in the Department of Electrical Engineering at the Indian Institute of Technology Jodhpur, Jodhpur, Rajasthan 342037 India. He was a researcher with the LS2N-IPI team (formerly IRCCyN-IVC), France, until 2016. His major research interests include the area of multimedia signal processing with a focus on perceptual aspects toward content capture, processing, and transmission. He is a Member of IEEE.

References

[1] M. Narwaria, "The transition from white box to black box: Challenges and opportunities in signal processing education," *IEEE Signal Process. Mag.*, vol. 38, no. 3, pp. 163–173, May 2021, doi: 10.1109/MSP.2021.3050996.

[2] G. Ras, N. Xie, M. van Gerven, and D. Doran, "Explainable deep learning: A field guide for the uninitiated," *J. Artif. Intell. Res.*, vol. 73, pp. 329–397, Jan. 2022.

[3] X. Bai et al., "Explainable deep learning for efficient and robust pattern recognition: A survey of recent developments," *Pattern Recognit.*, vol. 120, Dec. 2021, Art. no. 108102. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320321002892, doi: 10.1016/j.patcog.2021.108102.

[4] M. Zhang, H. You, P. Kadam, S. Liu, and C.-C. J. Kuo, "PointHop: An explainable machine learning method for point cloud classification," *IEEE Trans. Multimedia*, vol. 22, no. 7, pp. 1744–1755, Jul. 2020, doi: 10.1109/TMM.2019.2963592.

[5] P. Angelov and E. Soares, "Towards explainable deep neural networks (xDNN)," *Neural Netw.*, vol. 130, pp. 185–194, Oct. 2019, doi: 10.1016/j.neunet.2020.07.010.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[7] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, Aug. 2019, doi: 10.1109/JPROC.2019.2921977.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. 2016 IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[9] M. Tancik et al., "Fourier features let networks learn high frequency functions in low dimensional domains," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, Red Hook, NY, USA: Curran Associates, 2020, pp. 7537–7547.

[10] K. Zhang, L. Van Gool, and R. Timofte, "Deep unfolding network for image super-resolution," in *Proc. 2020 IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 3217–3226, doi: 10.1109/CVPR42600.2020.00328.

[11] E. Bjornson and P. Giselsson, "Two applications of deep learning in the physical layer of communication systems [Lecture Notes]," *IEEE Signal Process. Mag.*, vol. 37, no. 5, pp. 134–140, Sep. 2020, doi: 10.1109/MSP.2020.2996545.

[12] J. Ryu, M.-H. Yang, and J. Lim, "DFT-based transformation invariant pooling layer for visual classification," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 84–99.

[13] M. Cheung, J. Shi, O. Wright, L. Y. Jiang, X. Liu, and J. M. F. Moura, "Graph signal processing and deep learning: Convolution, pooling, and topology," *IEEE Signal Process. Mag.*, vol. 37, no. 6, pp. 139–149, Nov. 2020, doi: 10.1109/MSP.2020.3014594.

[14] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," 2020. [Online]. Available: http://arxiv.org/abs/1907.07374

[15] X. Li and F. Li, "Adversarial examples detection in deep networks with convolutional filter statistics," in *Proc. 2017 IEEE Int. Conf. Comput. Vis. (ICCV)*, pp. 5775–5783, doi: 10.1109/ICCV.2017.615.

[16] T. Spinner, U. Schlegel, H. Schfer, and M. El-Assady, "explAIner: A visual analytics framework for interactive and explainable machine learning," *IEEE Trans. Vis. Comput. Graphics*, vol. 26, no. 1, pp. 1064–1074, Jan. 2020, doi: 10.1109/TVCG.2019.2934629.

[17] J. Choo and S. Liu, "Visual analytics for explainable deep learning," *IEEE Comput. Graph. Appl.*, vol. 38, no. 4, pp. 84–92, Jul./Aug. 2018, doi: 10.1109/MCG.2018.042731661.

[18] Y. Xie, X. A. Chen, and G. Gao, "Outlining the design space of explainable intelligent systems for medical diagnosis," in *Proc. Joint ACM IUI Workshops co-Located 24th ACM Conf. Intell. User Interfaces (ACM IUI)*, C. Trattner, D. Parra, and N. Riche, Eds. Mar. 20, 2019, vol. 2327, pp. 1–7. [Online]. Available: http://ceur-ws.org/Vol-2327/IUI19WS-ExSS2019-18.pdf

SP

The system-centric approach typically attempts to approximate g via explicit characterization of different subsystems (components) of the physical process/application under consideration.

The focus on implicit modeling also raises the issue of lack of explainability/interpretability of the resultant DL-based mapping, or the black box problem.

What additional and practically useful insights can a system-centric approach reveal that can eventually help in the design of more transparent and explainable ML/DL systems?

We let S be a low-pass (moving average) filter and use it, as an example, to filter out signal information beyond 2 kHz.

Other practical aspects, such as controlling ripples in the passband or controlling gain in the transition band, can be explicitly handled in S.

In this context of on-device/on-edge computation, the black box nature can be a bottleneck in both DL model design and compression (reduction in the number of weights).

The black box problem may not necessarily be attributed to high dimensionality of the trained weights alone.

The system-centric approach can provide more flexibility toward explicitly analyzing the filtering characteristic of S if w is changed for any reason.