Computational experiments with cellular-automata generated images reveal intrinsic limitations of convolutional neural networks on pattern recognition tasks

Cite as: APL Mach. Learn. 2, 036102 (2024); doi: 10.1063/5.0213905
Submitted: 14 April 2024 • Accepted: 26 June 2024 •
Published Online: 15 July 2024

Weihua Lei,[1] Cleber Zanchettin,[2,3] Flávio A. O. Santos,[3] and Luís A. Nunes Amaral[1,2,4,a)]

AFFILIATIONS

[1] Department of Physics and Astronomy, Northwestern University, Evanston, Illinois 60208, USA
[2] Department of Chemical and Biological Engineering, Northwestern University, Evanston, Illinois 60208, USA
[3] Centro de Informática, Universidade Federal de Pernambuco, Recife, Pernambuco 52061080, Brazil
[4] Northwestern Institute on Complex Systems (NICO), Northwestern University, Evanston, Illinois 60208, USA

[a)] **Author to whom correspondence should be addressed:** amaral@northwestern.edu

ABSTRACT

The extraordinary success of convolutional neural networks (CNNs) in various computer vision tasks has revitalized the field of artificial intelligence. The out-sized expectations created by this extraordinary success have, however, been tempered by a recognition of CNNs' fragility. Importantly, the magnitude of the problem is unclear due to a lack of rigorous benchmark datasets. Here, we propose a solution to the benchmarking problem that reveals the extent of the vulnerabilities of CNNs and of the methods used to provide interpretability to their predictions. We employ cellular automata (CA) to generate images with rigorously controllable characteristics. CA allow for the definition of both extraordinarily simple and highly complex discrete functions and allow for the generation of boundless datasets of images without repeats. In this work, we systematically investigate the fragility and interpretability of the three popular CNN architectures using CA-generated datasets. We find a sharp transition from a learnable phase to an unlearnable phase as the latent space entropy of the discrete CA functions increases. Furthermore, we demonstrate that shortcut learning is an inherent trait of CNNs. Given a dataset with an easy-to-learn and strongly predictive pattern, CNN will consistently learn the shortcut even if the pattern occurs only on a small fraction of the image. Finally, we show that widely used attribution methods aiming to add interpretability to CNN outputs are strongly CNN-architecture specific and vary widely in their ability to identify input regions of high importance to the model. Our results provide significant insight into the limitations of both CNNs and the approaches developed to add interpretability to their predictions and raise concerns about the types of tasks that should be entrusted to them.

INTRODUCTION

Following the excitement about expert systems in the 1980s, the study of artificial intelligence (AI) entered another "winter" period.[1,2] The development of, first, deep neural networks and, then, convolutional neural networks (CNNs) and their success at computer vision tasks created a wave of interest in AI that has not yet subsided. CNNs have been reported to achieve super-human performance in object classification and face recognition.[3,4] Because of such successes, they have been applied to a variety of domains that have a strong influence on lives and the society, including detecting diseases from medical images,[5] self-driving cars,[6] job applications screening,[7] urban surveillance systems,[8] and scientific discovery.[9–11]

Some of the excitement and trust in CNNs may be due to claims of a strong analogy between how they work and how information is processed in the human brain. It is hypothesized, if not outright assumed, that CNNs develop representations of objects through their training.[12,13] Despite the successes and excitement, there is still a wide gap between the practical success and the theoretical understanding of CNNs.

Many studies have found that because CNNs are over-parametrized, they overfit the data and are susceptible to identifying "shortcuts," i.e., powerful predictors that are easy to learn but unrelated to the task at hand.[14–16] Consequently, those models can become very sensitive to small changes that are unremarkable to humans.[15,17] Concerns about the fragility of CNNs have led to the development of adversarial training strategies.[18,19] However, many studies have also demonstrated that adversarial training is not a panacea.[20–23]

The concern about the black-box nature of CNN models and their fragility has spurred the development of computational approaches to the interpretability of a model's predictions.[24–29] While there have been attempts at benchmarking attribution approaches (see Refs. 30–32 and the references therein), those attempts are limited by the difficulty in creating flexible, controllable, and rigorous datasets that are sufficiently large and require no human labeling.

Analyzing CNNs from a theoretical perspective is also not without challenges due to the astronomical degrees of freedom of the system. Some studies have suggested that the extraordinary performance of over-parametrized CNNs can be explained using renormalization group methods[33,34] or information theory perspectives.[35,36]

Attempts at addressing these heterogeneous challenges have also called upon experimental investigations. Such studies suggest that CNNs—unlike humans, who attended to the global shape of an object—preferentially learn "texture" instead of object shape.[37–39] However, progress is hindered because the available datasets are, to a great extent, convenience samples with uncharacterizable properties that make controlled experiments impossible.[40,41]

We present here a solution to the lack of flexible, controlled, rigorously characterizable datasets. Specifically, we use cellular automata (CA)[42,43] to generate images with rigorously controllable characteristics. This choice is motivated by the nature of images where the interactions between pixels determine the macroscopic pattern (rather than the value of individual pixels). The great success of CNNs has been attributed to their capabilities to capture the relations of multiple local pixels by convolutional operations.[44–46] CA offers a minimum toy example to the test of the above-mentioned hypothesis. In CA, macroscopic patterns arise from the interactions of a predefined number of pixels and a set of evolutionary rules. Moreover, CA allow for the definition of both extraordinarily simple and extraordinarily complex discrete functions and for the generation of limitless datasets of images without repeats. Cellular automata have long been an object of study in AI. Wolfram investigated the Turing universality of cellular automata,[42] and Crutchfield and Mitchell used them to study genetic algorithms.[43,47] More recently, Gilpin studied the equivalence between CNNs and two-dimensional CA,[48] and Mordvintsev *et al.* developed neural cellular automata for pattern regeneration.[49]

RESULTS

Cellular automata

Cellular automata are discrete functions of a finite number of discrete inputs—"rules"—that update the state of a system synchronously (see "Methods"). The most well-studied class of CA are denoted elementary cellular automata (ECA). They are applied to one-dimensional systems where each component's state is a binary variable and the state of each component is updated according to its state and those of its two nearest neighbors [Fig. 1(a)]. Despite their apparent simplicity, CA are capable of generating rich phenomena, including stationary, periodic, and chaotic patterns [see Figs. 1(b) and S1 for example of patterns].[42] Importantly, if one starts from a random state, a single ECA rule can generate an essentially limitless set of patterns—a system with $L = 100$ binary components can have $2^{100} \approx 1.3 \times 10^{30}$ distinct states.

Similarly, the number of distinct CA can be vast. An update rule that returns one of n_s possible values and takes as inputs its own state and that of its k nearest neighbors allows for the definition of $n_s^{n_s^{k+1}}$ distinct rules. For example, $k = 2$ and $n_s = 3$ yields $3^{3^3} \approx 7.6 \times 10^{12}$ distinct rules. Finally, and perhaps most importantly, CA build correlations within the local neighborhood of a pixel, thus generating objects with a shape. These characteristics make CA ideal generative processes for datasets of images used to probe how CNNs learn about the organization of the neighborhood of a given pixel.

Trainability of CNNs on CA-generated images

We first investigate the extent to which CNNs are able to learn CA rules [Fig. 1(a)]. To align with the classification nature of most vision tasks, we formulate our experiments as a classification problem. For each experiment, we first select the values of n_s and k and then select a rule $R(n_s, k)$ at random from the set of possible rules for those parameters and generate images by iterating R. For most experiments, we consider a state vector of length $L = 224$ and iterate the rule 223 times for each of 4000 + 1000 + 1000 randomly selected initial configurations for training, validation, and testing, respectively. We generate negative instances by shuffling the values in each of the CA-generated images (see "Methods"). Thus, the key difference between a positive and a negative CA-generated image will be the correlations within the neighborhood of a pixel.

We hypothesized that CNNs would readily learn to identify CA rules because of the similarity between the convolutional operations and the definition of local rules. Indeed, Gilpin showed that arbitrary CA may be represented by convolutional filters analytically.[48] We test the effectiveness of three state-of-the-art CNN architectures—VGG19,[46] ResNet18,[50] and GoogleNet[51]—in learning CA with different characteristics [Fig. 1(c)]. We find that there is little *qualitative* difference in the performance whether we fine-tune the fully connected layers of the three CNNs or retrain both features layers and the fully connected layers [Fig. 1(d)]. Thus, we conclude that these CNN architectures are highly effective at learning simple CA rules.

Learning limits of CNNs

Next, we systematically study the ability of CNNs to learn arbitrarily complex CA rules. To this end, we randomly select five CA rules $\{R_i(n_s, k)\}_{i=1,\ldots,5}$ for values of n_s in the range of 2–6 and values of k in the range of 2–20 [Fig. 2(a)]. For each R_i, we generate a dataset consisting of 4000 + 4000 images for training, 1000 + 1000 images for preventing over-fitting during training (validation), and 1000 + 1000 test images never seen during training for estimating the performance (testing). We detail the training

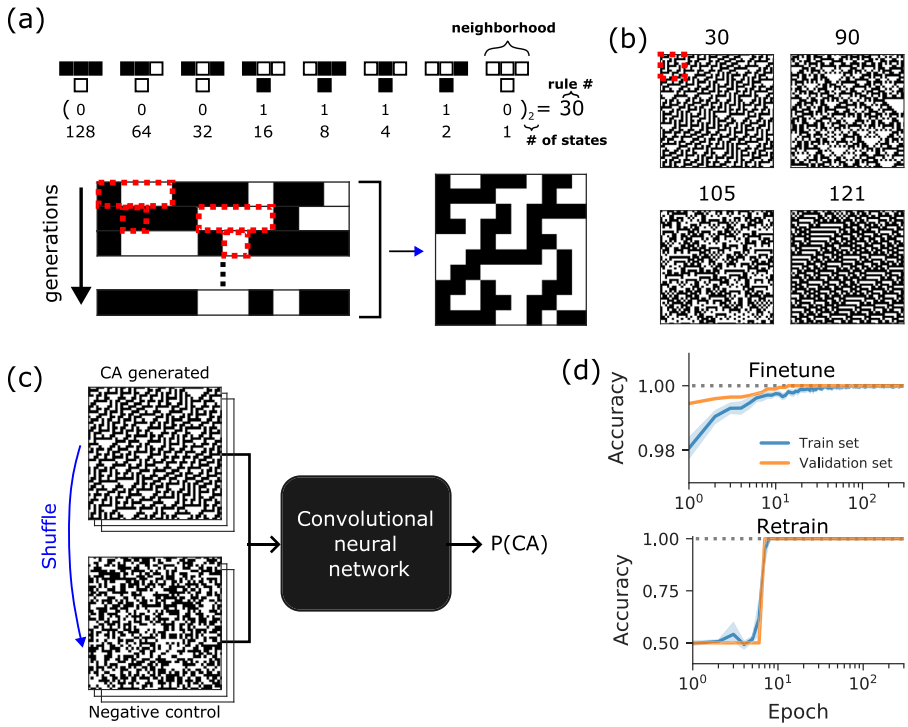

FIG. 1. Convolutional neural networks (CNNs) can easily learn to identify whether an image was generated by a cellular automaton. Cellular automata are sets of discrete functions, "rules," that update the state of a system synchronously. One-dimensional systems where each component's state is a binary variable and the state of each component is updated according to its state and those of its two nearest neighbors are denoted elementary cellular automata (ECA). (a) Illustration of ECA rule 30. The number assigned to the rule is just the number obtained by the binary output of the rule. Other, more complex, cellular automata allow for different numbers of states, n_s, and size of the neighborhood, $(1+k)$, where k is the number of nearest neighbors. We can use an ECA to generate an image by iterating the rule $(L-1)$ times for a system with L components with states assigned randomly. This generates an $L \times L$ image. $L = 10$ for the image in the figure. (b) Example outputs from four ECA rules illustrate the diverse outputs that can be generated by ECA (see Fig. S1 for more examples). (c) Illustration of the experimental setup. We generate a large number of images with $L = 224$ starting from random initial states and using a specific ECA rule. For each generated image, we create a control by shuffling the pixel values (negative sampling). Before feeding the set of images and controls to the CNN for training and validation, we stack three copies of the image to create an $(L, L, 3)$ array, as expected by the CNNs. We set up a classification task, in which the goal is to determine whether an image was generated using a certain rule or by a negative control. We study the performance of three state-of-the-art models—VGG19,[46] ResNet18,[50] and GoogleNet.[51] (d) Accuracy of the training and validation datasets for the first 300 epochs of training for VGG19 for rule 30. It is visually apparent that both transfer learning and retraining yield nearly perfect performance in fewer than 100 epochs.

process in the "Methods" section (also see Fig. S2 for learning curves).

We quantify the performance of a CNN on a given task as a linear transformation of the calculated accuracy. We denote this measure as the loss of predictability,[52]

$$\text{Loss} := 100\% \times \frac{1-A}{1-A_o}, \tag{1}$$

where A is the accuracy and $A_o = 0.5$ is the expected accuracy from random guessing in a balanced test. It is visually apparent that the three CNNs show decreasing performance as either the number of states or the number of neighbors increases [Figs. 2(a) and S4]. To rationalize these experimental results, we recall that the number of possible mappings for a CA characterized by parameters (n_s, k) increases as ${n_s}^{1+k}$. Following a statistical physics perspective, we can surmise that the latent space entropy for a given class of CA rules increases with the logarithm of the size of this latent space (see "Methods")

$$S = (1+k)\ \ln\ n_s. \tag{2}$$

This rationalization enables us to collapse the experimental data onto a single curve that shows a sharp transition from a learnable phase to an unlearnable phase as one increases the latent space entropy [Fig. 2(b)]. Those transitions suggest that there might be a complexity threshold beyond which CNNs cannot learn patterns in the data, leading to diminished performance as patterns become highly intricate. We model this transition with a standard generalized logistic function,

$$\text{Loss} = \frac{100\%}{1+e^{-(S-S_x)/w}}, \tag{3}$$

where S_x is the midpoint of the transition between learnable and unlearnable phases and w measures how broad the transition region

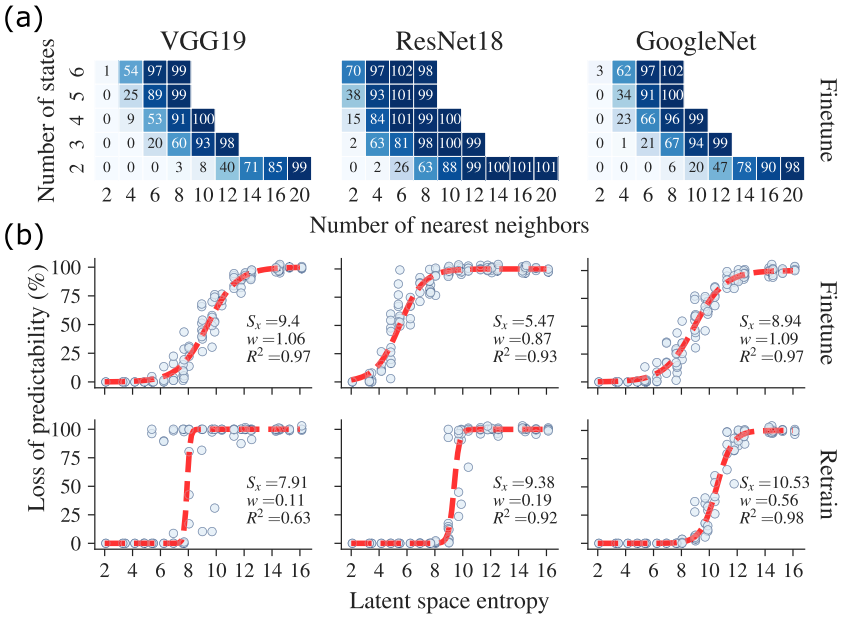

FIG. 2. CNN learning performance decreases sharply with task difficulty. We estimate the loss of predictability of different CNNs as we increase the number of k nearest neighbors affecting the update function and the number n_s of possible states for each component in the system. (a) For each (k, n_s) pair, we select five discrete functions at random (function numbers are listed in the supplementary material). We then generate 4000 + 4000 images and controls for the training dataset for each rule and another 2000 + 2000 for the testing dataset for each rule. Because of the vast space of possibilities ($2^{224} \gg 10^{50}$ for $n_s = 2$), an image will never be repeated in a set. We used transfer learning to train the CNN. The heatmaps show the average loss of predictability of transfer learning for VGG19, ResNet18, and GoogleNet on the testing datasets. (See Fig. S6 for heatmaps for retraining of the CNNs). As n_s and k increase, model performances sharply decrease. We can understand these results more fully by noting that the latent space entropy $S \propto k \ln n_s$. (b) Data collapse of loss of predictability vs entropy for fine-tuned and fully retrained models. If we interpret the entropy of the space of discrete functions considered as a measure of the difficulty in learning a particular function, then we can collapse the data onto a single function form. The data are shown as blue circles, and the lines are robust least square fits (see "Methods") to a generalized logistic function. This collapse is suggestive of a phase transition from learnable to unlearnable, similar to what was found for the P–NP transition.[53] Fully retraining the models yields sharper transitions at larger values of S, indicating a better model performance.

is. Our results suggest that retraining CNNs produces sharper transitions at higher complexities compared to fine-tune training.

While it is beyond the scope of this study to fully investigate the characteristics of this transition, we recognize its similarity to the transition reported between P and NP phases for some class of problems.[53–55] We also note that the dataset sizes used appear to be above the threshold needed for the CA rule to be learned if it was learnable (Fig. S5).

Significantly, latent space entropy may not be the only factor affecting the learnability of CA rules. To test this possibility, we systematically investigate the loss of predictability of VGG19 for every ECA rule. For each of the 256 ECA, we generate images of size $(50, 50, 3)$. As before, each dataset comprises an equal number of cellular automaton images and negative images—8000 images for training, 2000 images for selection, and 2000 images for testing.

We find that fine-tuned VGG19 shows different difficulties in learning 256 ECA (Fig. S1). For four rules belonging to the class of ECA rules capable of producing chaotic patterns,[56] we find a less than perfect performance for a fine-tuned model, whereas a fully retrained model yields a perfect performance in all cases (Fig. S3). This finding suggests that some learning situations may require full retraining of the CNN.

The simplest strongly predictive pattern is an attractor of the learning dynamics

It is well known that CNNs are prone to overfitting the training data and that they frequently learn "shortcuts" instead of task-related patterns (see the work by Wang[16] for an example of this). However, the field's understanding of the factors determining shortcut learning remains primarily qualitative and *ad hoc*. Armed with the wide spectrum of difficulty in novel cellular automata datasets, we are now able to design controlled experiments that test the characteristics of shortcut learning in CNNs.

Following on the work of Wang,[16] we first investigate the impact of signal destruction on a CNN's ability to learn the relevant pattern. Generalizing the approach in the work of Wang,[16] we generate datasets as before but shuffle pixel values outside of a central square taking a fraction f_s of the CA-generated image [Fig. 3(a)]. Our exploratory analysis confirms that for easy-to-learn patterns (CA rules drawn from small latent spaces), the CNN can learn the pattern even when it comprises a very small fraction, $f_s \to 0$, of the image. For higher complexity CA rules, the learning performance decreases more gradually with decreasing f_s, recapitulating the results in the work of Wang.[16]

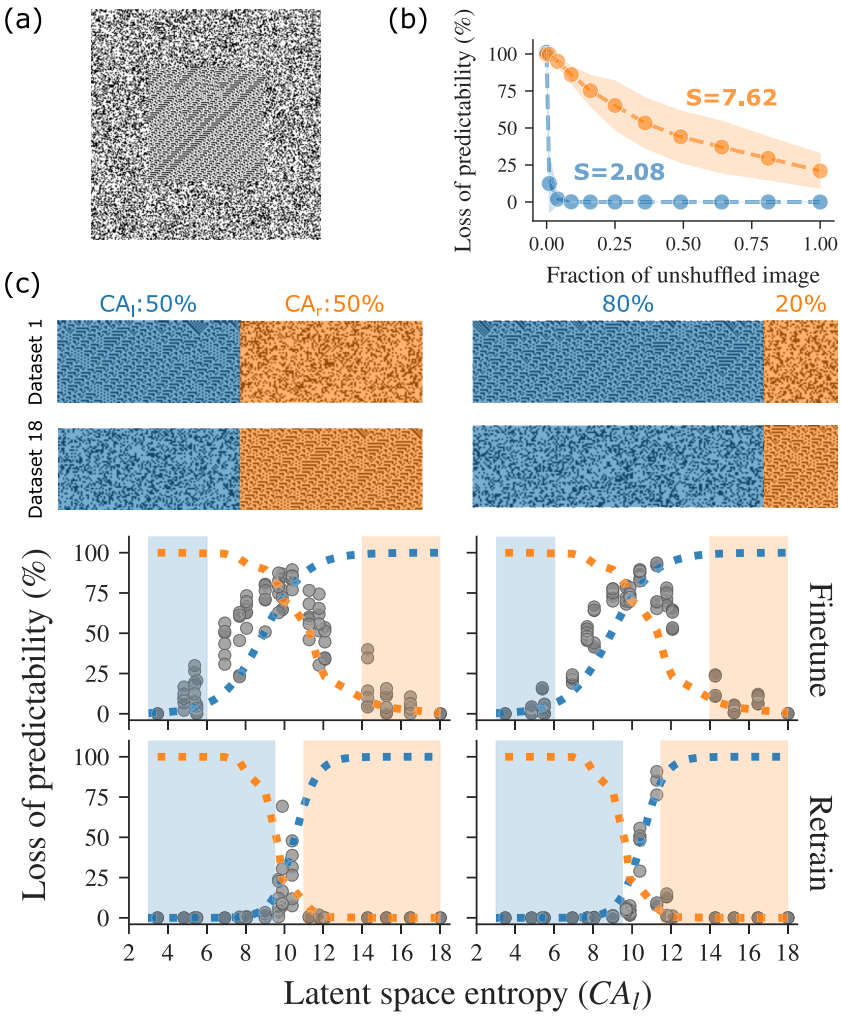

FIG. 3. Shortcut learning is unavoidable in CNNs. Here, we investigate the impact of both uninformative features and competition between patterns with different inherent levels of learnability. (a) Impact of uninformative features on the learning of CNNs. We generate two datasets of images and controls according to the process described earlier but then shuffled all but a fraction f_s of pixels in the center of the image. (b) Mean loss of predictability for GoogleNet models on two sets of automata with different latent space entropies (see Fig. S8 for other models). For the easy-to-learn automata (low latent space entropy), there is almost no loss of predictability until f_s approaches zero. In contrast, for the harder-to-learn—but still learnable—automaton, the loss of predictability increases steadily as f_s decreases. The shaded areas show one standard deviation bands. (c) Impact of competing predictive patterns. We create datasets in which a fraction f_l of the image is generated using a rule drawn from CA$_l$ and the remaining fraction is generated using a rule drawn from CA$_r$ (see the text and "Methods"). We consider two limiting cases. In the first, the image is split 50:50 between the two CA. In the second, the image is split 80:20. The blue dotted lines show the expected loss of predictability for learning the function applied on the left side of the image, and the orange dotted lines show the expected loss of predictability for learning the function applied on the right side of the image. The black circles show the actual loss of predictability calculated from our computational experiments for GoogleNet (see Fig. S8 for VGG19 and ResNet). The shaded regions show where learning is determined by the rule with the matching color. It is visually apparent that the CNN learns the simplest function for the extreme cases (blue-shaded and orange-shaded), but that when both rules have intermediate complexity, it does not learn either as well as if only the easiest-to-learn function was present. This situation is made more acute if the easiest-to-learn function occupies a smaller fraction of the image.

Next, we address the more realistic case of competition between multiple patterns that—if learned correctly—would yield high predictability but which would have different intrinsic levels of learning difficulty. Specifically, we generate patterns independently using pairs of CA with staggered latent space entropies and then concatenate the images. To specify the CA, we draw rules $\{R_i(n_s, k)\}_{i=1,\ldots,9}$ for values of n_s and k that yield a set of 18 latent space entropies (see Table S2). Then, if we denote by CA$_l$ (CA$_r$); the CA used to generate the leftmost (rightmost) columns of the image, CA$_l$ uses rules with increasing values of S, whereas CA$_r$ use rules with decreasing values of S. Again, images in the datasets are generated with sizes of (224, 224, 3), and each dataset comprises 8000 images for training, 2000 for validation, and 2000 images for testing.

If CNNs preferentially learn "shortcuts" (i.e., the easiest-to-learn strongly predictive pattern), then their performance should track the performance on the CA with the lowest latent space entropy. As the data shown in Fig. 3(c) demonstrate, this is indeed what happens but with a caveat. GoogleNet's performance on the entire image more closely tracks the expected performance for the CA with the lowest latent space entropy regardless of its location or of the fraction of the image it covers. However, when the CA have similar learning difficulties, GoogleNet displays a lower performance than what would be expected for the simplest CA to learn. We find similar behaviors for VGG19 and ResNet18 (Fig. S8) as well as for horizontally split images (Fig. S9). Thus, we conclude that the simplest strongly predictive patterns are unavoidable attractors of the training dynamics.

Benchmarking attribution methods

A criticism frequently thrown at CNNs is that they are uninterpretable "black boxes". In response, researchers have developed several approaches for estimating the importance of specific inputs

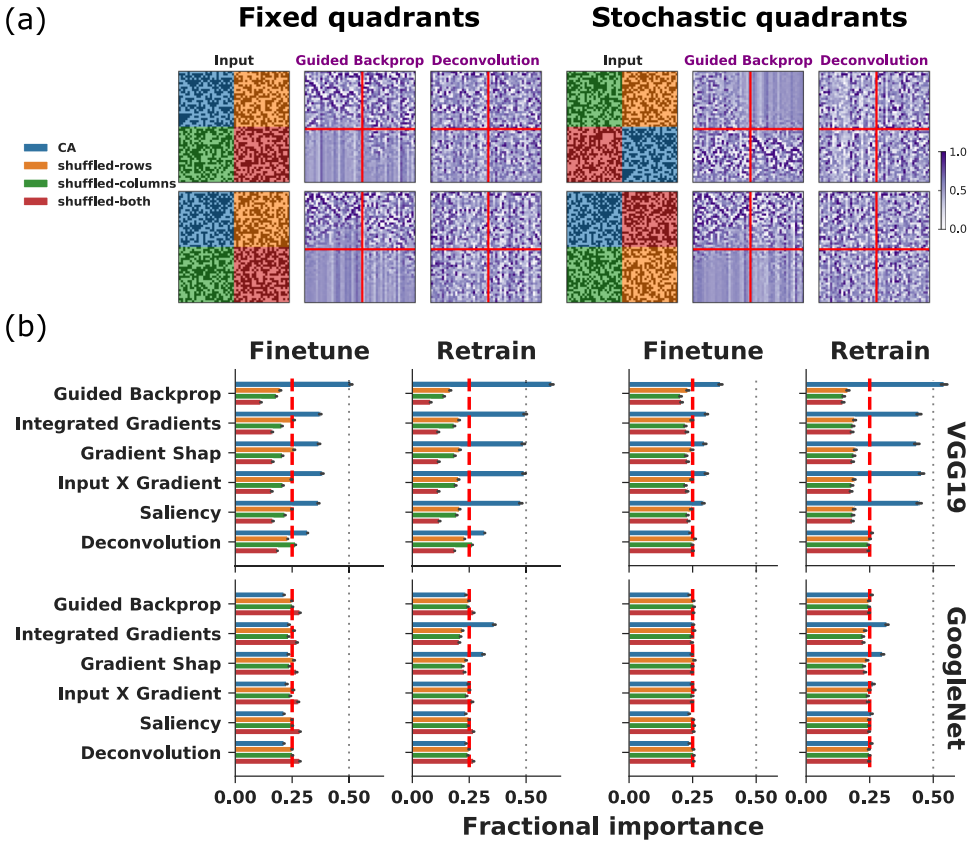

FIG. 4. CA-generated datasets enable the objective benchmarking of attribution methods. We investigate here the ability of different attribution methods to inform about the regions in an image that contribute most to a CNN's prediction. For concreteness, we focus on benchmarking attribution methods for VGG19 and GoogleNet. See the supplementary material for the results for other CNN architecture. We create datasets in which an image is generated using a specific CA but then divide each image into four quadrants. In one of the quadrants, we leave the image unaltered, but create specific negative controls within the other quadrants that we denote shuffled-rows, shuffled-columns, and shuffled-both (see "Methods"). In the first case (left side), we keep the treatment applied to each quadrant fixed. That is, the top left quadrant always remains unshuffled. In the second case (right side), each treatment is applied to a randomly selected quadrant. (a) ECA 90 generated images for fixed and stochastic treatments. This is an easy-to-learn rule and retrained VGG19 has high confidence $[P(\mathrm{CA}) \approx 100\%]$ in their classification. We display the attribution maps for two approaches—guided back-propagation and deconvolution (see the supplementary material for other approaches); it is visible that higher attribution values occur for the unaltered quadrant, followed by the quadrant with shuffled rows. We repeat this experiment for other ECA (see "Methods") with 8000 images for training and 13 ECA-generated images for calculating the attribution score. (b) Fractional importance estimated by different attribution approaches to quadrants assigned to different treatments when using different training approaches. The black error bars show the 95% confidence intervals for the estimation of the fraction. For fine-tuned models and for the case of fixed quadrant treatments, we find a systematic pattern of high importance for the unaltered quadrant, followed by the shuffled-row quadrant, then the shuffled-column quadrant, and finally the shuffled-both (uninformative) quadrant. Deconvolution is an exception. The results from fine-tuned models are more revealing for the stochastic quadrants treatment. Here, it is visible that only the unaltered quadrant provides higher than random importance (except, again, for deconvolution). By comparing estimated importance to expected importance, we can conclude that guided backpropagation performs the best, followed by saliency and input × gradient and that deconvolution performs only at the level of chance. When the models are fully re-trained, we see that the results follow our expectations more closely, even in the case of stochastic quadrants treatment. In fact, two of the methods even show S/N > 1 for the unaltered quadrants when applied to GoogleNet models.

to the CNN model. However, due to a lack of standard, rigorous, and controllable benchmarking datasets, we currently lack a deep qualitative and quantitative understanding of the strengths and weaknesses of each of those attribution methods. Fortunately, CA offer near limitless opportunities for altering inputs in controlled manners so as to truly measure the performance of competing attribution approaches and gain generalizable insights.

To rigorously quantify the performance of an attribution approach, we start by considering the 256 easy-to-learn ECA. We then alter the images created using a given ECA rule in order to destroy the pattern imposed by the rule. Specifically, we divide each ECA-generated image into four quadrants. In one quadrant, we leave unaltered; in a second (third) quadrant, we perturb by shuffling the sections of the columns (rows) falling within the quadrant; and in a fourth quadrant, we perturb by shuffling all the pixel values within the quadrant.

Our expectation is that during training, the CNN will assign the greatest importance to the inputs in the unaltered quadrant and the least importance to the inputs in the shuffled quadrant because the latter stores no information, whereas the former stores full information about the pattern. Similarly, because the rules are applied synchronously to an entire row, we expect that the CNN will assign greater importance to the quadrant with shuffled rows than to the quadrant with shuffled columns. These considerations effectively create sanity tests for attribution methods.[57]

We generate, according to each ECA rule, datasets of (50, 50, 3) images—8000 images for training, 2000 images for validation, and 2000 images for testing—and consider two treatments. In the first, which we denote with fixed quadrants, the positioning of the different types of perturbations is fixed. As shown in Fig. 4(a), the top left quadrant is where all the pixels are kept untouched, the top right is where rows are shuffled, the bottom left is where columns are shuffled, and the bottom right is where both rows and columns are shuffled.

Because in real-world conditions the patterns to be learned are not consistently within a fixed region of an image, we also consider a second treatment, which we denote stochastic quadrants. In this treatment, the positioning of the four types of perturbation is selected at random for each individual image [Fig. 4(c)].

We find that models still perform extraordinarily well for these datasets with test accuracies close to 100%. The question, thus, is how well different attribution methods work. Figure 4(a) shows two input images and the corresponding attribution maps obtained with guided backpropagation[28] and deconvolution[24] for VGG19 and GoogleNet (see the supplementary material for ResNet-18, other attribution methods, and other ECA rules). It is visually apparent that, unlike guided backpropagation, deconvolution cannot uncover the fact that the retrained VGG19 model must be finding different levels of information in different quadrants.

Figure 4(b) shows the average fractional importance of each region averaged over all the images in the datasets for fine-tuned and fully retrained VGG19 and GoogleNet models, and several attribution methods. Except for deconvolution, all the other attribution methods pass our sanity test for VGG19, but only two pass the sanity test for retrained GoogleNet and ResNet18 models (Figs. S10 and S11).

Next, we define a signal-to-noise ratio (S/N) using the fractional importance of the signal and shuffled-both quadrants, to quantitatively compare the performance of the different attribution methods. We find that several attribution methods have a strong performance (S/N ≈5) for VGG19, especially for the fully re-trained models.

The outcomes are more sobering for the stochastic quadrants treatment. First, we uncovered that fine-tuned VGG19 (pre-trained on ImageNet) has an inbuilt preference for features in the top left corner (see Fig. S12). Second, signal-to-noise ratios decrease across the board (but not as much for fully retrained models). For example, S/N approaches 1 for deconvolution, showing that it fails our sanity test.

For the other attribution methods, we see that either the CNNs are no longer able—or "willing" (because of shortcut learning)—to glean the partial information in the quadrants with shuffled rows or columns, or that the attribution methods are not sensitive enough to the differences in importance. In view of our results for shortcut learning (Fig. 3), we believe that learning from shuffled data is more challenging and thus less likely to occur, making the former explanation more likely.

We also extended this analysis to three local attribution methods—occlusion,[24] LIME,[58] and feature permutation.[59] All three fail our sanity tests (Fig. S10). In addition, our analysis confirms that attribution methods are highly architecture-dependent. Successful attribution methods in VGG19 fail to attribute high scores to the regions where the signal is located when used on other CNN architectures for both treatments (Fig. S11).

DISCUSSION

The lack of flexible, controlled, and rigorous benchmarking datasets has hindered our ability to understand the limitations of CNNs in tasks such as image classification. For the most part, we do not know whether there are limits to the complexity of the patterns that a CNN can learn, whether shortcut learning occurs only for certain types of patterns, or whether the interpretability of CNN predictions can be accomplished with the current attribution methods. This study significantly advances our understanding of all of these critical matters.

By using CA, a class of discrete functions operating on a finite number of discrete inputs, we can generate nearly limitless numbers of images displaying nearly limitless patterns. Significantly, by controlling the number of possible outputs of the function (with the parameter n_s) and the number of potential input combinations (with the parameters n_s and k), we can tune both the size of the neighborhood that the CNN has to learn to consider and the number of possible distinct patterns. For example, for CA that have three distinct outputs and consider the states of the two nearest neighbors ($S = 3.30$), the CNN has to learn to recognize the relevant 4-pixel combinations. CNNs accomplish this quite easily. However, the CNNs considered in this study start failing to learn when the latent space entropy approaches 10. Making the situation even more acute, not all $S \leq 10$ CA rules are likely to be equally learnable. It is unlikely that CNNs will be able to conquer all the problems that we pose to them in a task-appropriate manner.

This brings us to the problem of shortcut learning. Our results strongly suggest that CNN learning dynamics are attracted toward the simplest—i.e., that encompasses the smallest neighborhoods—strongly predictive pattern. This also explains why previous studies found CNNs biased toward the local texture

(fewer pixels) rather than the global shape of objects, which involves many pixels.[37,38] Without access to controlled experimentation or robust attribution methods, it is likely that one will not be able to determine whether a CNN's learning used a shortcut or not. This implies that uninterpretable CNNs are extraordinarily dangerous when used for critical matters. Indeed, studies have demonstrated that while CNNs appear to be able to accurately recognize faces, they show high errors for faces of individuals from marginalized groups,[60] while CNNs appear able to select good applicants from standard resumes, they incur unacceptable levels of false negatives for members of under-represented groups.[61]

While the use of attribution methods may ameliorate concerns about what a CNN is learning (see the remarkable study by DeGrave *et al.*[15]), this functionality relies on demonstrating that the attribution method is truly highlighting the inputs that contribute the most to the CNN's predictions. While prior studies have suggested that attribution should be used exclusively with CNN architectures for which they were developed, it has not been clear the extent to which some attribution methods may fail to appropriately identify important regions of an image. By perturbing CA-generated images, we are able to demonstrate that attribution methods optimized for a specific CNN architecture should not be used in conjunction with other CNN architectures and that attribution methods require retraining of the model—even those with high performance—in order to better capture important features for the model. For the attribution methods developed for VGG19, our results demonstrate beyond any doubt that local methods are utterly inadequate and that not all gradient methods are equally accurate.

Despite being task-specific, the insights gained from our study may extend to other areas of computer vision and beyond. They reveal the nature of model fragility, of learning limits in high entropy latent spaces, of the inevitability of shortcut learning, and of the limitations of current attribution methods that aim to make CNN predictions interpretable. Our study also demonstrates how synthetic datasets with rigorously quantifiable properties can advance our understanding of over-parameterized learning algorithms and open new opportunities for future research. Indeed, our approach to the generation of rigorously quantifiable datasets can be extended to other areas of computer vision, allowing greater insights into tasks such as segmentation and de-noising. Outside of computer vision, finite state automata have been used for modeling language production,[62,63] suggesting that the approach illustrated here may be extendable to that domain as well.

The strength of these conclusions must be tempered by recognizing some of the potential limitations of our study. First, it is not clear that CA can capture all the heterogeneities present in real-world photos. Nonetheless, CA can display extraordinarily complex and heterogeneous patterns that are able to demonstrate practical limits to the degree of complexity that a CNN can learn. Second, many of our experiments rely on CNNs pre-trained on ImageNet. These pre-trained models were primarily exposed to color natural images that differ quite dramatically from the black and white CA-generated images. We note, however, the similarity of the results obtained using fine-tuned vs retrained models increasing the robustness of our findings and the confidence one can have in the findings about the limitations of CNN learning. Indeed, while fully retrained models display higher performances for a range of intermediate pattern complexities, retraining does not remove the sharp transition from a learnable to an unlearnable phase. Third, we do not investigate how these results would change when using adversarial training approaches, an important open question that requires further study. Finally, our study focuses on CNNs and, thus, does not explore other approaches to deep learning, such as vision transformers.[64] Although alternative approaches are becoming increasingly popular and have demonstrated a strong performance in various computer vision tasks, much is still unknown about their ability to learn arbitrarily complex patterns, their interpretability, or their fragility. We believe that synthetic image generators, including CA, will also advance our understanding of the capabilities and limitations of those approaches.[64]

METHODS

CNN training

We use Pytorch (version 1.4.9)[65] and PyTorch Lightning (version 1.4.9)[66] implementations of the three CNNs. We train the models on a Tesla A100 GPU to minimize the cross entropy loss with a batch size of 256 and a learning rate of 10^{-4}.

For most of the results shown in the main text, we use VGG19, ResNet18, and GoogleNet models pre-trained on ImageNet and fine-tune them for each CA rule. Specifically, we fine-tune the fully connected layers with a learning rate equal to 0.0001 for 1000 epochs. During training, we monitor the validation loss and select the best model as the one with the lowest validation loss. We estimate the model performance on the test set, which is never seen by the model during the training.

For increased robustness of our conclusions, we also retrain the three CNNs. We initialize weights with the values from the pre-trained models but then train both the feature layers and the fully connected layers with a learning rate equal to 0.0001 for 1000 epochs. During training, we again monitor the validation loss and select the best model as the one with the lowest validation loss. We estimate the model performance on the test set, which is never seen by the model during the training.

Cellular automata

A transition rule of a cellular automaton with n_s distinct states, $X_s = \{0, 1, \ldots, s-1\}$, and k nearest neighbors is defined as the set of mappings from ${n_s}^{k+1}$ distinct permutations of an array of length $k+1$ each position with a value in X_s to another value also in X_s. Each set of mappings corresponds to one rule. For a particular set of parameters (n_s, k), n_s^{k+1} permutations each can map to n_s possible values; there are ${n_s}^{{n_s}^{1+k}}$ distinct mappings ("rules"). Elementary cellular automata (ECA) refer to the simplest among all possible nontrivial cellular automata, it comprises 256 different rules ($n_s = 2$ and $k = 2$).

Each rule $R(n_s, k)$ contains n_s^k mappings. For example, take ECA rule 30. The set of mappings are $R_{30}(2,3) = \{111 \mapsto \mathbf{0}, 110 \mapsto \mathbf{0}, 101 \mapsto \mathbf{0}, 100 \mapsto \mathbf{1}, 011 \mapsto \mathbf{1}, 010 \mapsto \mathbf{1}, 001 \mapsto \mathbf{1}, 000 \mapsto \mathbf{0}\}$. The number 30 is the decimal representation of the new states in a binary number system $[00\,011\,110]_2$.

We use the Python package CellPyLib (version 2.3.1)[67] to generate the CA datasets studied. Each image is generated by iterating a specific rule $L-1$ times for an array with length L and randomly

assigned initial values from X_s. We then stack the same $L \times L$ CA-generated image three times to get an array with three channels $(L, L, 3)$.

For instance, as shown in Fig. 1(a), the generation of the 10×10 array starts with an initial row of random values of length 10, [1 001 110 110]. The subsequent row, [1 111 000 100], is generated by applying transformation rules defined by $R_{30}(2, 3)$ under periodic boundary conditions. Each element in the second row is computed based on specific mappings from neighboring pixels in the preceding row: for example, the first element "**1**" is derived from pixels at positions 10, 1, and 2 in the first row (**010**), while the second "**1**" stems from pixels 1, 2, and 3 (**100**). This process iterates across the entire row, resulting in the formation of the second row [1 111 000 100]. This procedure is repeated sequentially eight more times to generate a 10×10 array. Subsequently, this 10×10 array is replicated three times to form a three-dimensional image of dimensions $(10, 10, 3)$.

Negative instance

To avoid introducing unintended differences that could act as "learning shortcuts," we shuffled the values in each of the CA-generated images to ensure the same probability distribution of states in CA-generated images and negative instances. In this way, the key difference between a CA-generated image and a negative instance is the correlation between pixel values within a local neighborhood.

Latent space entropy

Ludwig Boltzmann defined the entropy of a macroscopic system as an extensive, i.e., additive, measure of the number of its allowed microstates. Similarly, we define the latent space entropy S of a class of CA characterized by parameters (n_s, k) as

$$S = \ln \Omega, \tag{4}$$

where $\Omega = {n_s}^{1+k}$ is the number of mappings needed to specify a given rule. Thus, S measures the information needed to be encoded for a specific class of CA. For example, for the 256 ECA with $k = 2$ and $n_s = 2$, we have a latent space entropy $S = \ln 8 \approx 2.08$.

Shortcut learning experiments

To investigate the impact of uninformative features on the learning of CNNs, we generate two sets of rules of different latent space entropies ($S = 2.08$: $k = 2$ and $n_s = 2$, rules 8, 18, 4, 22, and 19; $S = 7.62$: $k = 2$ and $n_s = 2$, rules 1 021 279, 997 448, 1 058 637, 1 010 286, and 1 049 629).

To study the effect of competing predictive patterns, we generated datasets consisting of images where a fraction f_l was generated using a rule from CA_l, while the remaining $1 - f_l$ fraction was generated using a rule from CA_r. We generate 18 competing predictive patterns with an increasing (decreasing) latent space entropy—3.47, 4.83, 5.38, 5.49, 6.93, 7.69, 8.05, 9.01, 9.7, 9.89, 10.4, 11.27, 11.78, 12.08, 14.28, 15.25, 16.48, and 18.02 (reversed)—for CA_l (CA_r). We have listed the datasets in Table S2. To prevent shuffling information from the left and right sides together in negative controls, we generated CA_l, CA_r, and their corresponding negative controls independently and then concatenated them.

Robust least square

To get robust estimates of transition threshold S_x and avoid the impact of large fluctuations near the transition region (especially for retrained VGG19), we use robust regression to penalize outliers with large residues.[68] Robust least square curves in our results minimized the soft $L1$ loss, $\sum_i 2\sqrt{r_i^2 + 1} - 1$, instead of the $\sum_i r_i^2$ used in least squares, where $\{r_i\}$ are the residues.

Attribution methods

To create benchmark images for attribution methods, we divided each image into four quadrants. In one quadrant, we left the image unaltered, which we labeled as "CA". In one negative control region, we shuffled the rows within the quadrant ("shuffled-rows"), while we shuffled the columns within the quadrant ("shuffled-columns") in another. In the final region, we shuffled both the rows and columns within the quadrant ("shuffled-both").

The attribution methods evaluated in this study are implemented in Captum (version 0.4.0).[69] In both fixed quadrants and stochastic quadrants treatments, we calculated the fractional importance for 256 ECA rules with a test accuracy higher than 90%. For each of these ECA rules, we compute attribution scores for each attribution approach for a set of 25 images—13 generated by the CA model and 12 negative control—using batches of size 5. We then calculate the average over the high confidence $[P(\mathrm{CA}) \geq 0.9]$ CA-generated images (~1000 images) of the fractional importance for each quadrant as estimated by each attribution method.

For the hyperparameters of attribution methods, we use arrays of zeros as baselines for Integrated Gradients, Gradient SHAP, and occlusion. For Integrated Gradients specifically, we used the Gauss–Legendre algorithm with 200 time steps for the integration approximation. For Gradient SHAP, which requires random sampling to estimate gradients, we set the number of random samples to 5. Finally, for occlusion, the size of the patch for shading is set to 3×1.

SUPPLEMENTARY MATERIAL

The supplementary material 1.pdf includes the supplementary experimental results mentioned in the manuscript. The supplementary material 2.zip includes additional examples of attribution maps utilized for benchmarking attribution methods.

ACKNOWLEDGMENTS

This research was supported by the National Science Foundation Grant Nos. 1937123 and 2033604.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

W.L., C.Z., and L.A.N.A. conceived and designed the study. W.L. and F.A.O.S. performed the numerical simulations. W.L., C.Z., and L.A.N.A. performed the data analysis. W.L., C.Z., and L.A.N.A. created the figures. W.L., C.Z., F.A.O.S., and L.A.N.A. wrote, read, and approved the final version of the paper.

Weihua Lei: Conceptualization (equal); Formal analysis (equal); Investigation (equal); Methodology (equal); Validation (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal). **Cleber Zanchettin**: Conceptualization (supporting); Formal analysis (supporting); Investigation (supporting); Methodology (equal); Validation (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal). **Flávio A. O. Santos**: Conceptualization (supporting); Formal analysis (supporting); Investigation (supporting); Methodology (supporting); Validation (supporting); Visualization (supporting); Writing – original draft (supporting); Writing – review & editing (supporting). **Luís A. Nunes Amaral**: Conceptualization (equal); Formal analysis (equal); Funding acquisition (equal); Investigation (equal); Methodology (equal); Project administration (equal); Resources (equal); Supervision (equal); Validation (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The synthetic dataset is generated with codes in the repository https://github.com/amarallab/benchmarkCNNs.

The codes for reproducing the results in this article are available in the repository https://github.com/amarallab/benchmarkCNNs.

REFERENCES

[1] D. Crevier, *AI: The Tumultuous History of the Search for Artificial Intelligence* (Basic Books, New York, NY, 1993), ISBN: 9780465029976.

[2] M. Mitchell, *Artificial Intelligence: A Guide for Thinking Humans* (Farrar, Straus and Giroux, New York, 2019), ISBN: 9780374257835.

[3] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, in *2014 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, Columbus, OH, 2014), pp. 1701–1708, ISBN: 9781479951185, https://ieeexplore.ieee.org/document/6909616.

[4] K. He, X. Zhang, S. Ren, and J. Sun, in *2015 IEEE International Conference on Computer Vision (ICCV)* (IEEE, Santiago, Chile, 2015), pp. 1026–1034, ISBN: 9781467383912, http://ieeexplore.ieee.org/document/7410480/.

[5] G. Kaissis, A. Ziller, J. Passerat-Palmbach, T. Ryffel, D. Usynin, A. Trask, I. Lima, J. Mancuso, F. Jungmann, M.-M. Steinborn *et al.*, Nat. Mach. Intell. **3**, 473 (2021).

[6] Q. Rao and J. Frtunikj, in *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems* (ACM, Gothenburg Sweden, 2018), pp. 35–38, ISBN: 9781450357395, https://dl.acm.org/doi/10.1145/3194085.3194087.

[7] M. Mridha, R. Basri, M. M. Monowar, and M. A. Hamid, in *2021 International Conference on Science and Contemporary Technologies (ICSCT)* (IEEE, Dhaka, Bangladesh, 2021), pp. 1–6, ISBN: 9781665421324, https://ieeexplore.ieee.org/document/9642652/.

[8] X. Liu, W. Liu, T. Mei, and H. Ma, in *Computer Vision–ECCV 2016*, edited by B. Leibe, J. Matas, N. Sebe, and M. Welling (Springer International Publishing, Cham, 2016), Vol. 9906, pp. 869–884, ISBN: 9783319464749, 9783319464756, http://link.springer.com/10.1007/978-3-319-46475-6_53.

[9] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, Rev. Mod. Phys. **91**, 045002 (2019).

[10] N. Sapoval, A. Aghazadeh, M. G. Nute, D. A. Antunes, A. Balaji, R. Baraniuk, C. J. Barberan, R. Dannenfelser, C. Dun, M. Edrisi *et al.*, Nat. Commun. **13**, 1728 (2022).

[11] K. Choudhary, B. DeCost, C. Chen, A. Jain, F. Tavazza, R. Cohn, C. W. Park, A. Choudhary, A. Agrawal, S. J. L. Billinge *et al.*, npj Comput. Mater. **8**, 59 (2022).

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Commun. ACM **63**, 139 (2020).

[13] C. Olah, A. Mordvintsev, and L. Schubert, Distill (Nov. 7, 2017).

[14] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann, Nat. Mach. Intell. **2**, 665 (2020).

[15] A. J. DeGrave, J. D. Janizek, and S.-I. Lee, Nat. Mach. Intell. **3**, 610 (2021).

[16] D. Wang, J. Pers. Soc. Psychol. **122**, 806 (2022).

[17] J. Su, D. V. Vargas, and K. Sakurai, IEEE Trans. Evol. Comput. **23**, 828 (2019).

[18] I. J. Goodfellow, J. Shlens, and C. Szegedy, in 3rd International Conference on Learning Representations (ICLR), 2015; arXiv:1412.6572.

[19] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, in International Conference on Learning Representations, 2018, https://openreview.net/forum?id=rJzIBfZAb.

[20] H. Zhang, H. Chen, Z. Song, D. Boning, I. dhillon, and C.-J. Hsieh, in 7th International Conference on Learning Representations (ICLR), 2019, https://openreview.net/forum?id=HylTBhA5tQ.

[21] A. Nguyen, J. Yosinski, and J. Clune, in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Boston, MA, 2015), pp. 427–436, ISBN: 9781467369640, http://ieeexplore.ieee.org/document/7298640/.

[22] M. A. Alcorn, Q. Li, Z. Gong, C. Wang, L. Mai, W.-S. Ku, and A. Nguyen, in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Long Beach, CA, 2019), pp. 4840–4849, ISBN: 9781728132938, https://ieeexplore.ieee.org/document/8954212/.

[23] D. Hendrycks, K. Zhao, S. Basart, J. Steinhardt, and D. Song, in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Nashville, TN, 2021), pp. 15257–15266, ISBN: 9781665445092, https://ieeexplore.ieee.org/document/9578772/.

[24] M. D. Zeiler and R. Fergus, in *Computer Vision–ECCV 2014*, edited by D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars (Springer International Publishing, Cham, 2014), Vol. 8689, pp. 818–833, ISBN: 9783319105895, 9783319105901, http://link.springer.com/10.1007/978-3-319-10590-1_53.

[25] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje, "Not just a black box: Learning important features through propagating activation differences," Proc. Machine Learn. Res. **70**, 3145–3153 (2017).

[26] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, in *2017 IEEE International Conference on Computer Vision (ICCV)* (IEEE, Venice, 2017), pp. 618–626, ISBN: 9781538610329, http://ieeexplore.ieee.org/document/8237336/.

[27] S. M. Lundberg and S.-I. Lee, in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)* (Curran Associates, Inc., Red Hook, NY, 2017), pp. 4768–4777, ISBN: 9781510860964.

[28] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," arXiv:1412.6806 [cs] (2015).

[29] Y. Sun and M. Sundararajan, in *Proceedings of the 12th ACM Conference on Electronic Commerce* (ACM, San Jose, CA, 2011), pp. 177–178, ISBN: 9781450302616.

[30] M. Yang and B. Kim, "Benchmarking attribution methods with relative feature importance," arXiv:1907.09701 [cs, stat] (2019).

[31] R. Tomsett, D. Harborne, S. Chakraborty, P. Gurram, and A. Preece, in *Proceedings of the AAAI Conference on Artificial Intelligence* (AAAI, 2020), Vol. 34, pp. 6021–6029, https://ojs.aaai.org/index.php/AAAI/article/view/6064.

[32] H. Shah, P. Jain, and P. Netrapalli, in *Advances in Neural Information Processing Systems* (NeurIPS, 2021), Vol. 34, p. 2046.

[33] P. Mehta and D. J. Schwab, "An exact mapping between the variational renormalization group and deep learning," arXiv:1410.3831 [cond-mat, stat] (2014).

[34]D. A. Roberts, *The Principles of Deep Learning Theory: An Effective Theory Approach to Understanding Neural Networks* (Cambridge University Press, New York, 2022), ISBN: 9781316519332.
[35]N. Tishby and N. Zaslavsky, in *2015 IEEE Information Theory Workshop (ITW)* (IEEE, Jerusalem, Israel, 2015), pp. 1–5, ISBN: 9781479955244, 9781479955268, http://ieeexplore.ieee.org/document/7133169/.
[36]A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox, J. Stat. Mech.: Theory Exp. **2019**, 124020.
[37]R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, in International Conference on Learning Representations, 2019, https://openreview.net/forum?id=Bygh9j09KX.
[38]N. Baker, H. Lu, G. Erlikhman, and P. J. Kellman, PLoS Comput. Biol. **14**, e1006613 (2018).
[39]L. Scimeca, S. J. Oh, S. Chun, M. Poli, and S. Yun, "Which shortcut cues will DNNs choose? A study from the parameter-space perspective," arXiv:2110.03095 [cs, stat] (2022).
[40]H. Hosseini, B. Xiao, M. Jaiswal, and R. Poovendran, in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)* (IEEE, Cancun, Mexico, 2017), pp. 352–358, ISBN: 9781538614181, http://ieeexplore.ieee.org/document/8260656/.
[41]S. Madan, T. Henry, J. Dozier, H. Ho, N. Bhandari, T. Sasaki, F. Durand, H. Pfister, and X. Boix, Nat. Mach. Intell. **4**, 146 (2022).
[42]S. Wolfram, *A New Kind of Science* (Wolfram Media, Champaign, IL, 2019), ISBN: 9781579550257.
[43]M. Mitchell, P. T. Hraber, and J. P. Crutchfield, "Revisiting the edge of chaos: Evolving cellular automata to perform computations," Complex Syst. **7**(2), 89–130 (1993).
[44]K. Fukushima, Biol. Cybern. **36**, 193 (1980).
[45]Y. L. Cun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, and D. Henderson, *Handwritten Digit Recognition with a Back-Propagation Network* (Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1990), pp. 396–404, ISBN: 1558601007.
[46]K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations* (ICLR, San Diego, CA, 2015).
[47]M. Mitchell, J. P. Crutchfield, R. Das *et al.*, "Evolving cellular automata with genetic algorithms: A review of recent work," in *Proceedings of the First International Conference on Evolutionary Computation and its Applications (EvCA'96)*, edited by E. K. Goodman (Presidium of the Russian Academy of Sciences, Moscow, Russia, 1996).
[48]W. Gilpin, Phys. Rev. E **100**, 032402 (2019).
[49]A. Mordvintsev, E. Randazzo, E. Niklasson, and M. Levin, Distill **5**(2), e23 (2020).
[50]K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2016), pp. 770–778.
[51]C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Boston, MA, 2015), pp. 1–9, ISBN: 9781467369640, http://ieeexplore.ieee.org/document/7298594/.
[52]W. Lei, C. Zanchettin, Z. E. Ho, and L. A. Nunes Amaral, APL Mach. Learn. **1**, 046118 (2023).
[53]K. Xu and W. Li, Sci. China, Ser. E: Technol. Sci. **42**, 494 (1999).
[54]C. Moore, arXiv:1702.00467 (2017).
[55]D. Gamarnik, C. Moore, and L. Zdeborová, J. Stat. Mech.: Theory Exp. **2022**, 114015.
[56]S. Wolfram, Physica D **10**, 1 (1984).
[57]Y. Zhou, S. Booth, M. T. Ribeiro, and J. Shah, in *Proceedings of the AAAI Conference on Artificial Intelligence* (AIAA, 2022), Vol. 36, pp. 9623–9633, https://ojs.aaai.org/index.php/AAAI/article/view/21196.
[58]M. T. Ribeiro, S. Singh, and C. Guestrin, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, San Francisco, CA, 2016), pp. 1135–1144, ISBN: 9781450342322, https://dl.acm.org/doi/10.1145/2939672.2939778.
[59]C. Molnar, *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*, 2nd ed. (Christoph Molnar, Munich, Germany, 2022), ISBN: 9798411463330.
[60]J. Buolamwini and T. Gebru, in *Proceedings of the 1st Conference on Fairness, Accountability and Transparency. Proceedings of Machine Learning Research (PMLR)*, edited by S. A. Friedler and C. Wilson (PMLR, 2018), Vol. 81, pp. 77–91 https://proceedings.mlr.press/v81/buolamwini18a.html.
[61]J. Dastin, *Ethics of Data and Analytics* (Auerbach Publications, 2018), pp. 296–299.
[62]J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 3rd ed. (Pearson/Addison Wesley, Boston, 2007), ISBN: 9780321455369.
[63]C. Moore and S. Mertens, *The Nature of Computation* [Oxford University Press, Oxford (England); New York, 2011], ISBN 9780199233212.
[64]A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16 × 16 words: Transformers for image recognition at scale," arXiv:2010.11929 [cs] (2021).
[65]A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, in *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (Curran Associates, Inc., Red Hook, NY, 2019).
[66]W. Falcon *et al.* (2019). "PyTorch lightning," GitHub. https://github.com/PyTorchLightning/pytorch-lightning
[67]L. Antunes, J. Open Source Software **6**, 3608 (2021).
[68]P. J. Huber, Ann. Math. Stat. **35**, 73–101 (1964).
[69]N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, B. Alsallakh, J. Reynolds, A. Melnikov, N. Kliushkina, C. Araya, S. Yan *et al.*, "Captum: A unified and generic model interpretability library for PyTorch," arXiv:2009.07896 (2020).