nature computational science

ARTICLES

https://doi.org/10.1038/s43588-021-00132-w

Explainable neural networks that simulate reasoning

Paul J. Blazek[1,2,3] and Milo M. Lin[1,2,3,4] ✉

The success of deep neural networks suggests that cognition may emerge from indecipherable patterns of distributed neural activity. Yet these networks are pattern-matching black boxes that cannot simulate higher cognitive functions and lack numerous neurobiological features. Accordingly, they are currently insufficient computational models for understanding neural information processing. Here, we show how neural circuits can directly encode cognitive processes via simple neurobiological principles. To illustrate, we implemented this model in a non-gradient-based machine learning algorithm to train deep neural networks called essence neural networks (ENNs). Neural information processing in ENNs is intrinsically explainable, even on benchmark computer vision tasks. ENNs can also simulate higher cognitive functions such as deliberation, symbolic reasoning and out-of-distribution generalization. ENNs display network properties associated with the brain, such as modularity, distributed and localist firing, and adversarial robustness. ENNs establish a broad computational framework to decipher the neural basis of cognition and pursue artificial general intelligence.

Cognitive theories can be implemented in and tested by artificial intelligence (AI) systems. Models inspired by psychology are called 'symbolic' because they explicitly encode our experiences of such cognitive processes as reasoning, perception and language. However, symbolic AI ignores the neural basis of cognition and is limited to specialized or simple tasks because of its need for manual feature curation[1–3]. On the other hand, 'connectionist' models simulate the activity of interconnected neurons, treating cognitive processes as emergent phenomena arising from firing patterns distributed across a neural network[4]. Connectionist AI is exemplified by deep learning approaches, which have recently demonstrated superhuman accuracy across a large range of tasks, utilizing artificial neural networks composed of layers of neurons loosely inspired by biology and trained by backpropagation and stochastic gradient descent[1,3,5,6].

Despite their successes, current deep neural networks face fundamental limitations that restrict their usage and suggest they do not accurately reflect the neural basis of cognition. They are widely regarded as black boxes because, currently, there is no mechanistic or quantitative explanation for how the distributed activities of individual neurons give rise to network outputs or cognitive experiences, especially in the deeper layers responsible for mapping features to outputs[7–11]. Deep learning models fail to replicate symbolic manipulation and other fundamental cognitive processes such as high-level reasoning and deliberation. For example, deep learning cannot generalize out-of-distribution, instead requiring large labeled datasets that must contain samples from the same distribution as the target task[3,9,12,13]. Deep neural networks, furthermore, are infamously fooled by small, human-imperceptible input perturbations called adversarial attacks, emphasizing that they process information differently than the brain[8,9].

Although artificial neurons were originally inspired by biological neurons[14] and convolutional layers were inspired by visual processing in the retina and visual cortex[3], many characteristic population-level properties of the brain are not typically found in deep neural networks. They lack, especially in deeper dense layers, both the modular connectivity[15,16] and the sparse, localist firing specific for certain stimuli[17–21] found in the brain. It has also proven difficult to understand and mimic the rapid, local neuroplastic changes in the brain responsible for dynamic cognitive processes[22,23].

To understand the neural basis of cognition and pursue artificial general intelligence, it is necessary to bridge the divide between symbolism and connectionism. Progress has been limited, despite substantial efforts to develop theory, improve and expand architectures, optimize training, and mix and match problem-specific techniques[3,7,9]. Some speculate that connectionist AI could simulate symbolic reasoning only by accounting for more complexities of neurobiology, such as biochemistry, spike trains, massive networks and neuroplasticity[24].

In this Article, we develop a simple yet scalable neurocognitive model of neural information processing in which each neuron is a specialized decision-making agent within a hierarchical network. To demonstrate that this model is sufficient to encode complex cognitive capabilities, we implemented it in a general-purpose machine learning algorithm for classification tasks that builds deep neural networks that are explainable, capable of symbolic manipulation, and better reflect neurobiological properties of the brain. These networks perform well on standard AI benchmarks and surpass existing deep neural networks on tasks involving reasoning and out-of-distribution generalization.

Results

Directly encoding cognitive operations in neural networks. Because a given neuron will fire for certain stimuli and not others, its activity represents a distinction between these two sets of possible inputs. We will focus on neural distinctions as minimally sufficient for cognition, abstracting away the details of biochemistry, neural dynamics and plasticity. In our model, each neuron makes one of two types of distinction: absolute or relative. A 'concept neuron' makes an absolute distinction between a specific subset of

[1]Green Center for Systems Biology, University of Texas Southwestern Medical Center, Dallas, TX, USA. [2]Department of Bioinformatics, University of Texas Southwestern Medical Center, Dallas, TX, USA. [3]Department of Biophysics, University of Texas Southwestern Medical Center, Dallas, TX, USA. [4]Center for Alzheimer's and Neurodegenerative Diseases, University of Texas Southwestern Medical Center, Dallas, TX, USA. ✉e-mail: milo.lin@utsouthwestern.edu

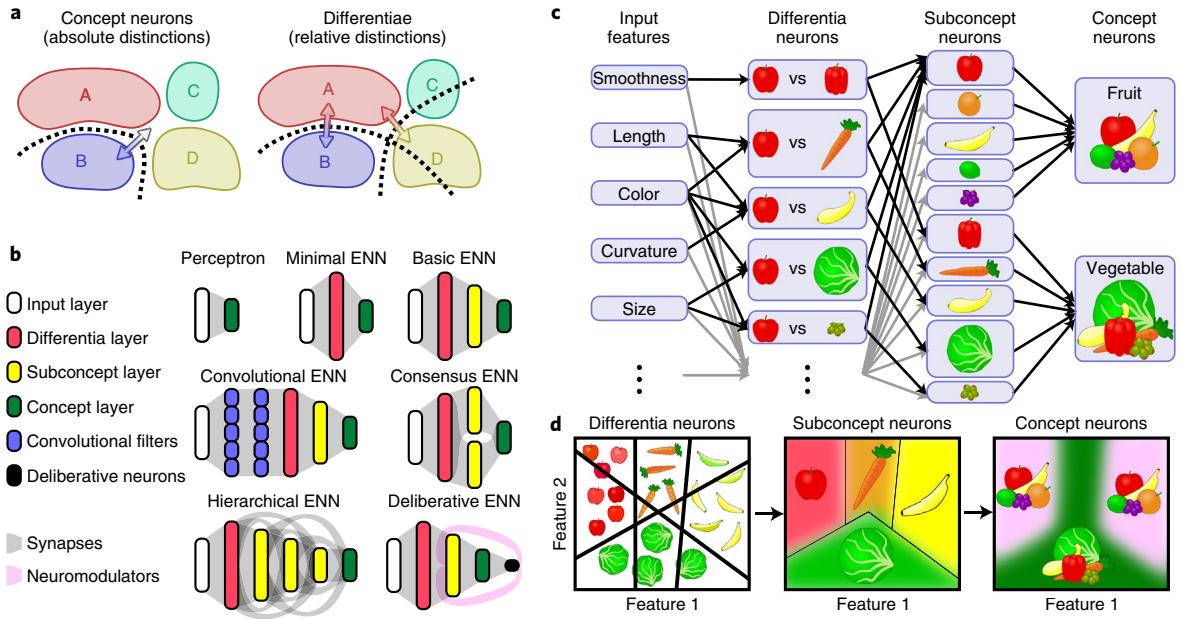

Fig. 1 | Connectivity principles of ENNs. a, Concept neurons absolutely distinguish a single concept (for example, B-like or not-B-like), while differentia neurons relatively distinguish pairs of concepts (for example, more A-like versus more B-like). **b**, The many ways to design ENN architectures. **c**, A detailed examination of the structure of a basic ENN showing how its architecture directly integrates reasoning processes (for example, recognizing culinary types of produce). Only apple-distinguishing differentia neurons are depicted here, and each differentia neuron connects to its corresponding subconcept neurons, which then connect to the corresponding concept neuron. **d**, The structure of conceptual space is learned directly by ENNs. Differentia neurons form hyperplane decision boundaries (lines) in conceptual space. They feed forward to subconcept neurons, each forming a subregion (colored areas) defined by differentia neuron boundaries. These feed into concept neurons, each forming a possibly disconnected conceptual region from its subconcepts.

similar stimuli (that is, a concept) and all other possible stimuli (for example, 'like B' versus 'not like B' in Fig. 1a). A 'differentia neuron' instead makes a relative distinction between stimuli from two different concepts (for example, the dotted line separating 'more like A' versus 'more like B' in Fig. 1a). Note that this A-versus-B differentia neuron still outputs whether other stimuli (such as from C or D) are more A-like or B-like. In the brain, these specialized distinctions could, in principle, be performed redundantly by populations of neurons, for example to provide robustness.

Often, a concept may be impossible to absolutely distinguish with a single neuron (for example, in Fig. 1a, B would not be linearly distinguishable), so it must use the outputs of multiple upstream differentia neurons to distinguish it. Furthermore, concept neurons may also receive inputs from related concept neurons, allowing them to integrate the hierarchical structure of concepts. This is especially important for complex, non-convex concepts (for example, words that have multiple definitions) that require division into subconcepts[25].

These two connectivity principles can be summarized as (1) neurons make either relative or absolute distinctions between concepts and (2) networks integrate related distinctions hierarchically. These principles set our neurocognitive model apart from existing connectionist models such as deep neural networks, where neurons do not make interpretable distinctions and information is entangled across neural populations.

This model is consistent with theories from cognitive psychology and philosophy. Differentia and concept neurons are defined by the decision boundaries they make relative to inputs[26], while the ability to recognize specific subconcepts is consistent with prototype and exemplar theories[26,27] as well as localist models[17–19]. Furthermore, philosophy of mind has long described cognitive processes like reasoning and perception as series of distinctions[28–32]. For example, in the frameworks of Aristotle and Thomas Aquinas, a thing's 'essence' is its definition, which first identifies its superordinate genus and then distinguishes it within the genus using qualities called 'differentiae'[30,33–36]. As reasoning is an iterative process of making judgments (that is, distinctions) to arrive at conclusions[31,37–39], neural systems that incorporate our proposed principles are implementing a form of reasoning.

Constructing artificial neural networks that reason. To test whether this model is sufficient to simulate reasoning in neural systems, we used it to construct artificial neural networks that we call essence neural networks (ENNs). Because an artificial neuron's output is the weighted sum of its inputs passed through a nonlinear activation function, it makes a hyperplanar distinction[40]. This is useful, because hyperplanes can separate disjoint convex regions, which conceptual space theory identifies as simple, natural concepts[2,41].

Our principles permit considerable freedom in ENN architectural design and learning algorithm. Figure 1b shows different classes of simple, layered ENN architectures. Basic ENNs (Fig. 1c) (1) receive inputs, (2) pass them to all of the differentiae, (3) determine similarity to subconcepts and then (4) choose an optimal output concept. Appropriate neural connectivity can also be directly inferred from the structure of conceptual space (Fig. 1d). Differentia neurons form the initial distinctions, and downstream subconcept neurons form subregions bounded by differentia hyperplane distinctions. Concept neurons form the final regions by unifying upstream subconcept subregions (note that fuzzy distinctions produce curved decision boundaries). We will show that basic ENNs can solve a wide range of machine learning tasks, as well as describe more complex architectures that increase the scope and performance of ENNs.

We developed a machine learning algorithm that trains ENNs for arbitrary classification problems (Methods). For basic ENNs, it first takes training samples from each concept (that is, each target label) and divides them into subconcepts. Differentia neurons are generated by computing linear support-vector machines (SVMs) between each pair of subconcepts. Subconcept neurons are generated using

Table 1 | Performance of the ENN compared to GDNs

Problem	Training samples	Output classes	Input size	Conv. Layers	Layer 1 size	Layer 2 size	GDN time (min)	ENN time (min)	GDN accuracy	ENN accuracy
Perception tasks:										
Rectangles	50,000	2	28×28	—	201	56	2.6	7.5	99.98%	99.78%
MNIST	60,000	10	28×28	—	394	60	1.7	5.4	98.35%	97.27%
MNIST	60,000	10	28×28	6(5×5), 16(5×5)	127	84	6.1	12.4	98.70%	97.65%
MNIST	60,000	10	28×28	32(3×3), 64(3×3)	3167	84	12.7	16.0	98.93%	99.14%
CIFAR-10	50,000	10	28×28×3	60(3×3), 120(3×3)	3333	108	52.4	275	62.25%	64.78%
Symbolic reasoning tasks:										
Logic gates	64	2	18	—	4	4	0.06	0.01	100%	100%
Orientation	56	2	28×28	—	784	56	0.20	0.05	Diagonals: 63.3% Outlines: 70.7% Dotted: 62.0% Zigzags: 68.1%	Diagonals: 100% Outlines: 100% Dotted: 100% Zigzags: 100%
MAX-SAT	3,800	2	100×20	—	599	400	353	1.9	MAX-3SAT: 3.61 less MAX-SAT: 4.49 less	MAX-3SAT: 0.10 more MAX-SAT: 0.08 more
MAX-SAT	99,000	2	500×100	—	2999	2000	—	2316	—	MAX-3SAT: 1.34 more MAX-SAT: 1.35 more
TSP	90	10	55	—	405	90	1.15	0.075	2.04 longer	0.00 shorter
BDTs	20	10	1024	—	180	20	0.05	0.02	0.79 more	0.001 fewer

The results shown are the results of training an ENN and a GDN of the same size on several datasets (including 3 different network architectures for MNIST and 2 MAX-SAT datasets of different sizes). The following are shown for each problem: the number of training samples, number of output target classes, input dimensions, convolution filter number and size, number of neurons in each layer, GDN and ENN training times, and GDN and ENN accuracy on the test set. Accuracy on NP-hard problems is compared to standard greedy algorithms (Methods); units are clauses for MAX-SAT, map units for TSP (traveling salesman problem), tree depth for BDTs (binary decision trees).

SVMs to separate each subconcept from the others using differentia neuron outputs as inputs. During this stage, less important differentiae are pruned away. Finally, concept neurons are generated using SVMs to separate each concept from all the others using subconcept neuron outputs.

ENN training is fundamentally different from the current optimization-based approach to deep learning, which instead minimizes a loss function via backpropagation and stochastic gradient descent[3,9,40]. We trained ENNs and compared them to standard gradient-descent-trained networks (GDNs) of the same size (Methods) on both perception-like tasks (for example, the MNIST (Modified National Institute of Standards and Technology) and CIFAR-10 (Canadian Institute For Advanced Research) datasets) and symbolic tasks (for example, MAX-SAT). Even without deliberate code optimization, ENN training times were practical and convenient (Table 1 and Supplementary Fig. 1), and ENN accuracy was comparable to that of size-matched GDNs on benchmark perception tasks (Table 1 and Supplementary Fig. 1). Yet, the purpose of this work was not to improve performance on existing benchmarks, but rather to develop a general-purpose neural network model that can simulate symbolic cognition in an interpretable manner. Therefore, we show that ENNs differentiate themselves from GDNs by excelling in explainability, symbolic reasoning (Table 1) and robustness.

The explainable structure and function of ENNs. ENNs are inherently explainable in function (each neuron makes a specific distinction) and structure (weights and biases are designed for optimal distinctions). To demonstrate explainability, we analyzed ENNs trained on the MNIST dataset of 70,000 images of handwritten digits (Fig. 2a)[5], a popular choice for assessing machine learning algorithms. We also tested it on a synthetic dataset with horizontally or vertically oriented rectangles (Supplementary Fig. 3a and Methods). GDNs show minimal intelligible structure in the weights between image pixels and first-layer neurons (Fig. 2c and Supplementary Fig. 3c), while ENN differentia neuron weights are easily interpretable because they are designed via linear SVMs. This can be appreciated visually: differentiae positively weight pixels associated with a particular subconcept and negatively weight those of a different subconcept (Fig. 2b,c and Supplementary Fig. 3b,c).

Connections between deeper layers of neurons in GDNs are typically even less interpretable, with no sparsity[15] or modularity[16]. In ENNs, each subconcept neuron only requires inputs from its associated differentiae (such as in Fig. 1c). In general, they are free to be connected to all differentia neurons, yet we observed that subconcept neurons rely heavily on just these associated differentiae (Fig. 2d,e and Supplementary Fig. 3d,e). To analyze modularity, we split the positive and negative outgoing connections for each neuron between a separate excitatory and inhibitory neuron. Each split neuron was assigned to a group based on which output was most affected by the neuron's firing (Methods). Individual ENN neuron firing had a much greater impact on the network output than did GDN neurons (Supplementary Fig. 4). Sorting each connectivity matrix by group demonstrates the ENN's modularity (Fig. 2d,e), with a measurable difference between weights within-group

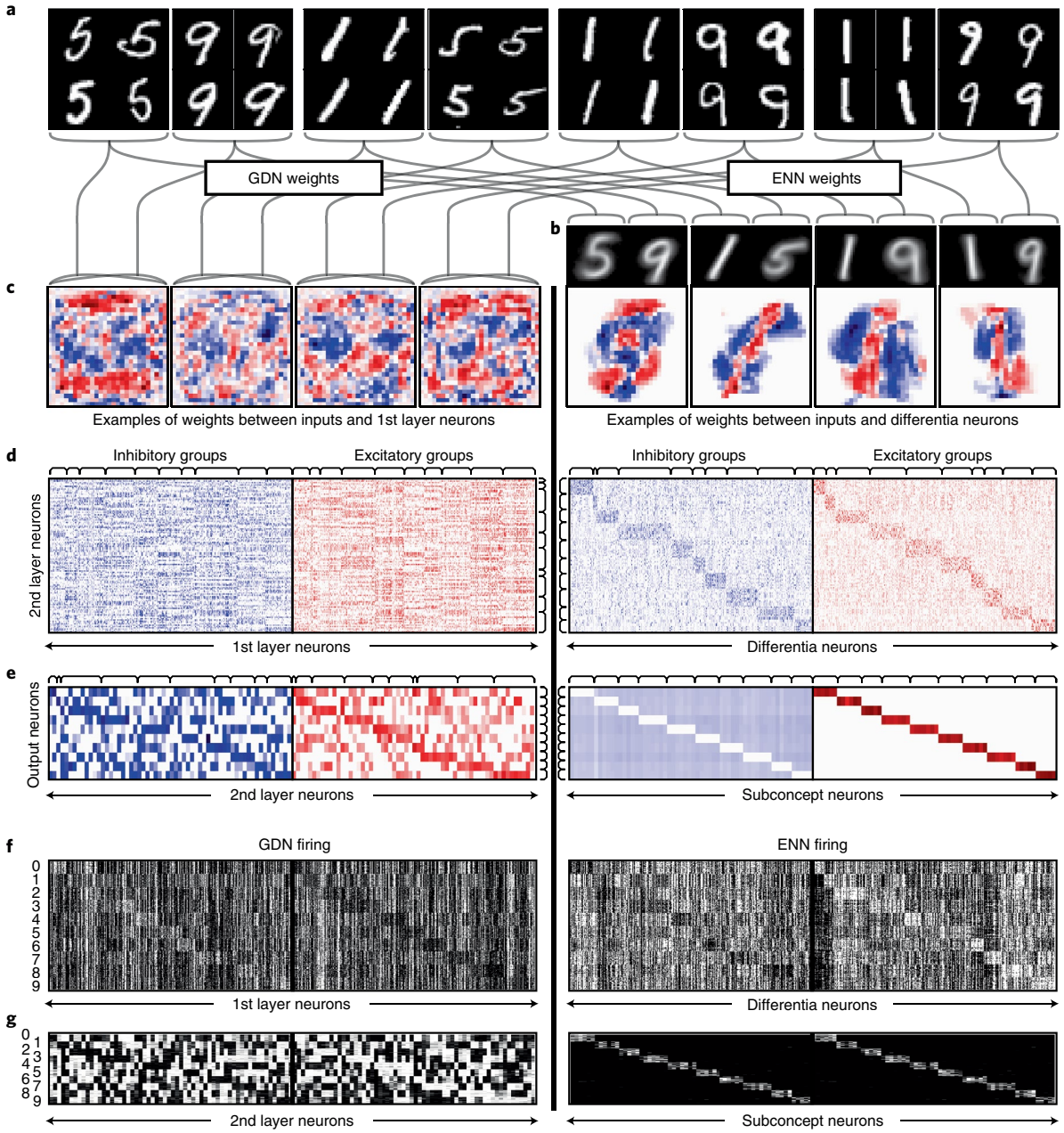

Fig. 2 | The explainability of ENN neural structure and firing. a, Example MNIST training images. **b**, ENN learning first divides images within each output concept into subconcepts. Averages from representative subconcepts are shown, with lines connecting individual images to their subconcept. **c**, First-layer neurons, showing synaptic weights from each input pixel to the neuron (red are positive weights, blue are negative). The ENN neurons shown are those that distinguish the subconcept pairs in **b**. For comparison, the GDN neurons shown are those that maximally separate the same subconcept pairs. **d**,**e**, Matrices of synaptic weights between the first and second layers of neurons (**d**) and the second and third layers (**e**), split into inhibitory and excitatory neuron pairs and sorted into functional groups. **f**,**g**, The firing rates of first-layer (**f**) and second-layer (**g**) neurons on test images from each class (white is maximal firing, black is no firing).

versus between-group (Supplementary Fig. 4a,b). This is consistent with the brain's modular architecture and axonal organization into neural tracts. By contrast, GDN connectivity is unstructured until the output layer, and even that is much less modular than ENNs (Supplementary Fig. 4a).

Furthermore, ENNs hierarchically separate distributed and localist firing patterns, with distributed firing in the differentia layer (Fig. 2f) and localist firing in subconcept and concept layers (Fig. 2g). GDN hidden layers display no localist firing. These results show that the biologically observed segregation of distributed and localist firing may be a signature of hierarchically segregated relative and absolute neural distinctions.

ENN reasoning analysis and flexible designs. *Error analysis.* Explainability of function permits explanation of both correct predictions (Supplementary Fig. 5) and errors, which is necessary for ethical and legal considerations and for preventing rare catastrophic failures. For example, sometimes we can identify differentia neurons that are single-handedly responsible for producing an error on MNIST (Fig. 3a). These were used to identify features missing from

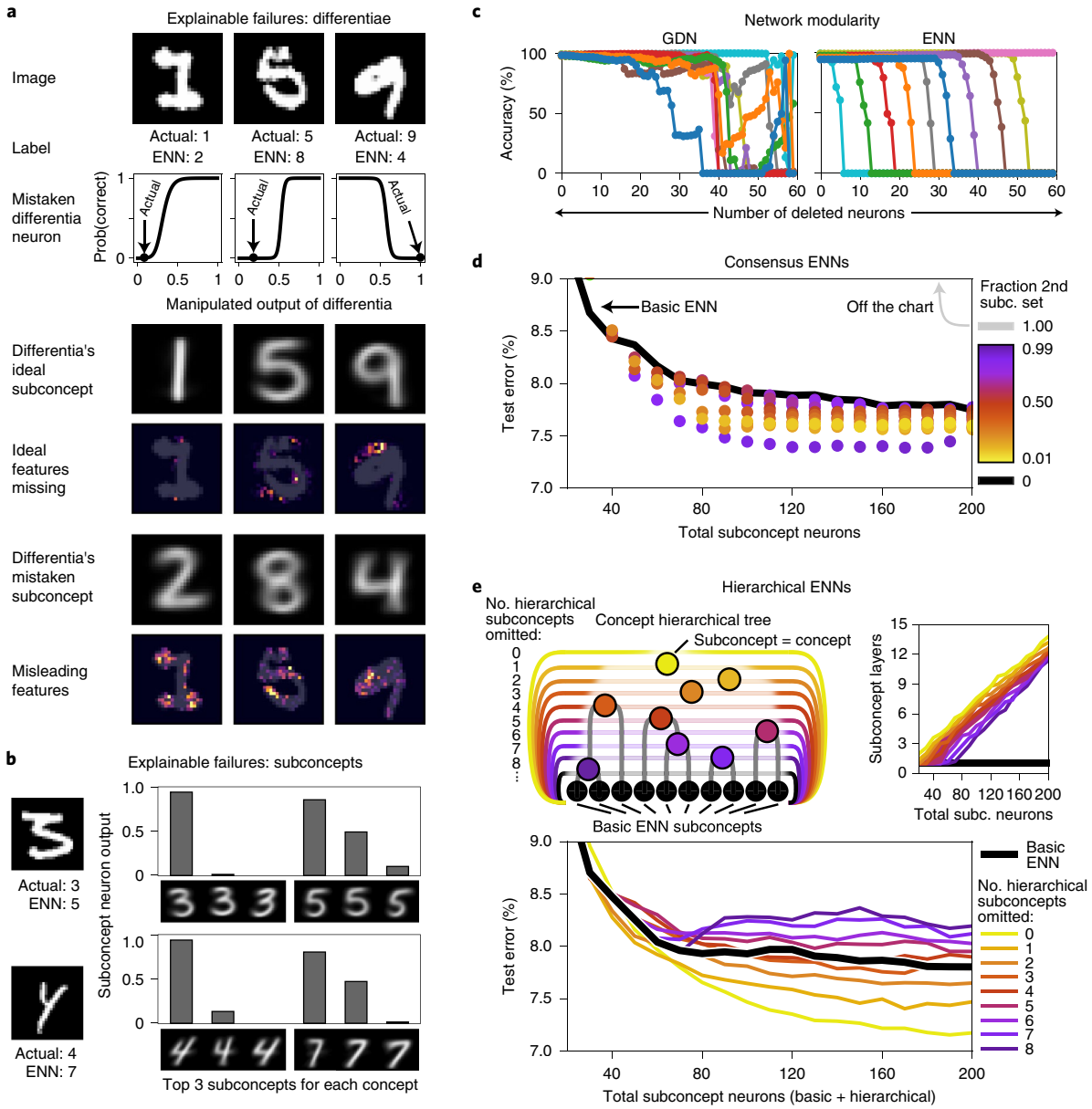

Fig. 3 | Structural analysis and flexibility of ENNs. a, Examples of images mislabeled by the ENN for which a differentia neuron is single-handedly responsible for misclassification. Also shown are features missing from the image compared to the differentia's ideal subconcept and the features that misled the ENN in making its mistake. **b**, Some mislabeled images had optimal firing on a correct subconcept neuron that was drowned out by several lesser-firing incorrect subconcept neurons. **c**, The GDN's and ENN's accuracies in classifying each of the 10 classes of digits, colored separately, as a simulated lesion grew by sequentially deleting subconcept neurons. **d**, The test error of consensus ENNs as the fraction of subconcept neurons coming from the second, unsupervised set is varied from 0 (that is, all basic ENN subconcepts, black) to 1 (that is, all unsupervised concepts, gray). **e**, Hierarchical ENNs can use the full concept hierarchical tree or omit a given number of parent nodes, leading to a different number of subconcept layers and different test errors.

the image that could have prevented the error, as well as misleading features that caused the error (Fig. 3a). For other errors, the image optimally activated a correct subconcept neuron, but multiple erroneous subconcept neurons with lower outputs combined to outweigh the correct subconcept (Fig. 3b).

The inherent modularity of ENNs allows a network to be drastically pared down yet retain select concepts. When we progressively deleted subconcept neurons, the ENN lost accuracy for specific output classes one at a time, while GDNs demonstrated an unpredictable, non-monotonic loss of function (Fig. 3c). The ENN's sequential loss of specific functions is reminiscent of progressive neurological disorders with focal lesions, such as multiple sclerosis[42] and vascular dementia[43].

ENN-based network architectures. Explainability of structure permits flexibility in designing more sophisticated architectures (Fig. 1b). For example, we have proposed one rudimentary way to train convolutional ENNs (Methods). Features are learned from subimages, then locally connected subconcept neurons serve as convolutional filters (Supplementary Fig. 2). At the end of the convolutional layers, a basic ENN is trained to produce the final output, yielding accuracy similar to size-matched convolutional GDNs (Table 1).

An ENN can be designed to explicitly integrate multiple independent perspectives to form a stronger consensus. A simple example of a consensus ENN has multiple overlapping sets of subconcepts learned independently of each other (Fig. 1b), which we implemented by learning a second subconcept set fully unsupervised (Methods). For ENNs of fixed size, varying the ratio of the number of subconcepts from this second overlapping set improved the performance over basic ENNs (Fig. 3d). (These consensus ENNs were trained on 20 sets of only 1,000 random MNIST images to better illustrate the improvement.)

ENNs can also have deeper subconcept layers that integrate the hierarchical organization of concepts (Fig. 1b and Methods). Because our basic ENNs learn subconcepts via hierarchical clustering, each node of the concept hierarchical tree going to the root can be used to generate higher-level subconcept neurons (Fig. 3e). These hierarchical subconcept neurons were placed in deeper layers, receiving both direct and skip connections from differentia neurons and previous subconcept layers. Depending on how many of these hierarchical subconcepts were omitted ('0 omitted' meaning a full hierarchical ENN), we found that hierarchical ENNs had better performance than basic ENNs, even when controlling for network size (again training on 20 sets of 1,000 MNIST images).

In addition to different neural architectures, ENN explainability allows for dynamic changes to its learned parameters, similar to neuroplastic and neuromodulatory changes in biological neurons. After the ENN was fully trained, we could optionally add a neuromodulatory feedback element that is active when no output probabilities exceed 50%, dynamically increasing or decreasing firing thresholds (that is, bias factors) of subconcept neurons (Fig. 1b and Methods). This neuromodulation simulates the slow, deliberative reasoning of dual-process-theory's system 2, in contrast to the rapid, intuitive reasoning of system 1 and standard deep neural networks[44]. Deliberative ENNs offered some improvement in classification accuracy, especially when training with less data (Supplementary Fig. 1b), and were necessary for some symbolic reasoning tasks.

These and other possible architectures may prove useful when applying ENNs to more complex problems in the future. For example, we show that adding convolutional filters, deeper layers using hierarchical subconcepts, and a second set of consensus subconcepts increased ENN performance compared to that of size-controlled GDNs on the CIFAR-10 natural image dataset (Table 1)[45,46]. These initial results do not make use of the more advanced architectures and techniques necessary for GDNs to obtain state-of-the-art performance (for example, data augmentation, regularization and model averaging), indicating ENNs have similar potential for improvement.

Learning algorithms that generalize across domains. ENNs are explainable because each neuron performs the symbolic task of computing similarity to opposing concepts. Symbolic representation is necessary for reasoning, and discrete symbols are particularly useful and straightforward to analyze. We therefore tested whether ENNs are sufficient to simulate discrete symbolic reasoning by using neurons with outputs of only 0 or 1 (or 0.5 for ties), which is similar to spike-based neural coding[47] but not possible for GDNs because there is no gradient. Symbolic reasoning is particularly important for learning rules from instructive examples and applying them to different but related problems. This has been called single domain generalization, in which a model trained on a single distribution of samples generalizes to unseen target distributions without additional training[48]. This has not been possible using standard deep learning without training on at least some data from the target distribution[9,48–50]. Going even further, we tested whether ENNs could generalize out-of-distribution from simple examples to complex tasks.

We trained a symbolic ENN and size-matched GDN to distinguish vertical and horizontal orientations of shapes in an image, using just 56 28×28 images containing only a full-length vertical or horizontal white stripe (Fig. 4a). The symbolic ENN is explainable, meaning we could readily translate its weights into equivalent pseudocode (Supplementary Text), suggesting symbolic ENNs could be used for automatic code generation. We assessed how well the networks generalized to several target test sets, including tens of thousands of diagonal line segments, rectangles, dotted lines and zigzag lines (Fig. 4b). Strikingly, unlike the GDN, the ENN performed perfectly on all of them (Fig. 4c). We tried optimizing GDNs by augmenting with convolutional filters and choosing network architectures with the best test set performance, but still could not find GDNs that transferred well (Fig. 4c and Supplementary Fig. 6). To see how difficult it is for gradient descent to perform single domain generalization, we added random perturbations to the symbolic ENN weights and used these as the initial parameters for GDN training. Gradient descent could not train a network that generalized well once these weight perturbations exceeded 1–3% (Supplementary Fig. 7).

Generalizing from simple problems is the basis for designing greedy algorithm heuristics such as for NP-hard problems, of which one of the best studied is the Boolean satisfiability (SAT) problem[51]. MAX-SAT is a classic generalization of SAT that asks for the maximum number of satisfiable clauses in a Boolean formula in conjunctive normal form[52]. We developed training sets in which each formula has only one non-empty clause containing at most two literals (Fig. 4d and Methods), with the target network output being the ideal Boolean assignment for the next unassigned variable. We tested the trained networks on two target datasets with different numbers of non-empty clauses, one for MAX-3SAT, in which clauses have exactly three randomly chosen literals, and another for the more general MAX-SAT, with clauses of random lengths (Fig. 4e and Methods). Both the GDN and ENN were trainable on the 10-variable, 100-clause training set, but we were unable to train GDNs on larger formulae in a reasonable time, probably due to the extreme non-overlapping sparsity of the training samples. On MAX-3SAT and MAX-SAT, only the ENN generalized well (Fig. 4f). We were again able to translate the ENN weights into equivalent pseudocode (Supplementary Text), which is similar to human-designed algorithms. It is more complex than the pure greedy algorithm, which assigns TRUE to x_1 when (x_1) is more prevalent than (not x_1) and vice versa for FALSE; instead the ENN is more similar to the MAX-SAT greedy algorithm with the best approximation ratio of 3/4[53]. Surprisingly, the ENN performed better than both algorithms on random MAX-3SAT and MAX-SAT, a performance difference that grew for ENNs trained and tested on progressively larger formulae (Fig. 4f).

We also trained symbolic ENNs on two other NP-hard problems, the traveling salesman problem (TSP) and the optimal binary decision tree (BDT) problems[51,54], which both required the deliberative architecture (Methods). Analysis of the ENNs and translation to pseudocode revealed that, on the TSP, the ENN discovered the nearest-neighbor greedy heuristic, while on the BDT problem, the ENN learned an algorithm with similar performance to the standard Classification and Regression Trees (CART) algorithm (Supplementary Fig. 8 and Supplementary Text). Together, these results demonstrate that symbolic ENNs can learn human-understandable algorithms from simple instances of a task that generalize out-of-distribution to complex instances.

Robustness of ENNs to input noise and adversarial attacks. Symbolic reasoning is more robust because it makes clear-cut distinctions. We tested this by tasking a GDN and a symbolic ENN to concurrently learn all 16 two-input Boolean functions (for example, AND, OR, NAND, XOR) by training on all 64 entries of the truth

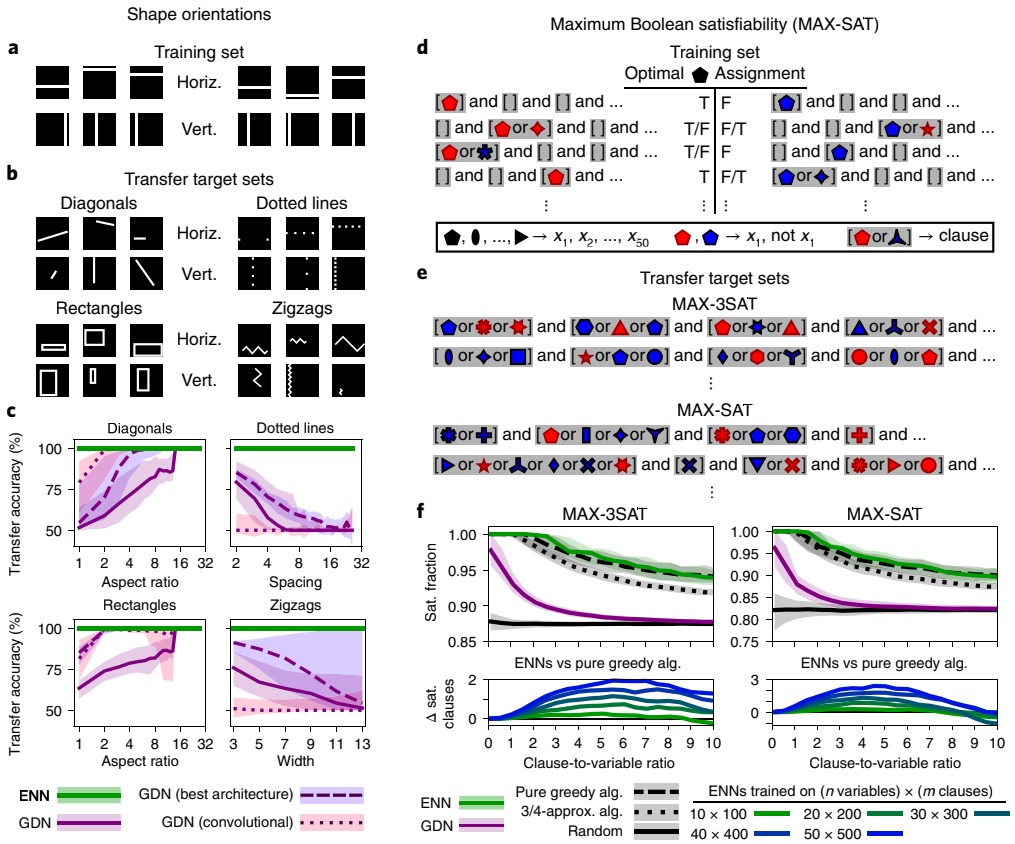

Fig. 4 | Transferring algorithms from simple to complex problems. a–c, Orientation problem results. **a**, The training set contains images with a perfectly horizontal or vertical image-spanning stripe. **b**, The transfer target sets include four types of shape whose target labels are determined by their length-to-height ratio. **c**, Although the ENN generalized perfectly to the transfer sets, GDNs could not. Median accuracies at different difficulty levels are presented, with shading showing interdecile ranges. **d–f**, MAX-SAT results. **d**, Examples of formulae in the training set, displaying empty clauses except one containing x_1 and sometimes another variable. Each is labeled as TRUE (T), FALSE (F) or both with a preference. **e**, The target sets had many clauses, with MAX-3SAT using three literals per clause and MAX-SAT having a variable amount. **f**, For formulae of maximum size 10×100, the expected value and interquartile range (shading) of the fraction of satisfiable clauses are shown for varying numbers of clauses, as determined by the ENN, GDN, random assignment, pure greedy or 3/4-approximation greedy heuristics. Shown beneath are the differences between ENNs of various sizes compared to the pure greedy algorithm.

table (Fig. 5a). ENN weights were easily interpreted and directly converted to a logic circuit (Fig. 5b). Although both the GDN and ENN achieved perfect training accuracy (Table 1), visualizing the decision boundaries showed that only the ENN made regular intuitive distinctions (Fig. 5c) when tasked to interpolate between training data on fuzzy inputs.

To measure the separation between inputs and decision boundaries on higher-dimensional problems, we took individual test set images and interpolated either toward an image of a different class or toward white noise. Along these trajectories we found the network's nearest decision boundary and measured the normalized L_1 distance (that is, the average pixel difference) from the starting images. We found that both GDNs and ENNs similarly space decision boundaries between images (Fig. 5d and Supplementary Fig. 9a). However, when interpolating between images and white noise, we observed ENN decision boundaries spaced at greater distances than those of GDNs, indicating more robust decision boundaries. This resulted in a greater tolerance to input noise by ENNs than GDNs (Fig. 5e and Supplementary Fig. 9b).

A robust decision boundary arrangement is particularly important when defending against adversarial attacks. We generated adversarial images against GDNs and ENNs using the fast gradient sign method[8] and measured the minimum perturbation ($\epsilon_{\min}$) needed for each image to fool its network. Because adversarial images transfer well between networks[8], we also tested each network on adversarial images designed against the other. Not only were ENNs several-fold more robust to self-adversarial images than were GDNs, but ENNs also were not easily fooled by adversarial attacks designed against GDNs (Fig. 5f,g and Supplementary Fig. 9c,d). On MNIST, we found that adversarial images designed against the ENN surprisingly fooled the GDN more than the ENN. Furthermore, these differences in robustness were even greater for larger networks (Supplementary Fig. 9e–h). Interestingly, ENN adversarial perturbations also appeared more interpretable (Supplementary Fig. 10).

Discussion

Basic ENNs are most similar in design to Voronoi neural networks (VNNs)[55], which use Voronoi tessellation to learn hyperplane separations between all individual training points and then combine them with AND and OR gates (thus implementing one-nearest-neighbor classification). One key distinction is that ENNs learn aggregate concepts and subconcepts rather than memorizing the training set, allowing ENNs to scale to large, complex training sets without suffering from exponential scaling. While sharing similar goals with other hierarchical AI models of the cortex and of reasoning[14,56], ENN training is automatic, flexible, scalable and useful for both perception and symbolic reasoning tasks.

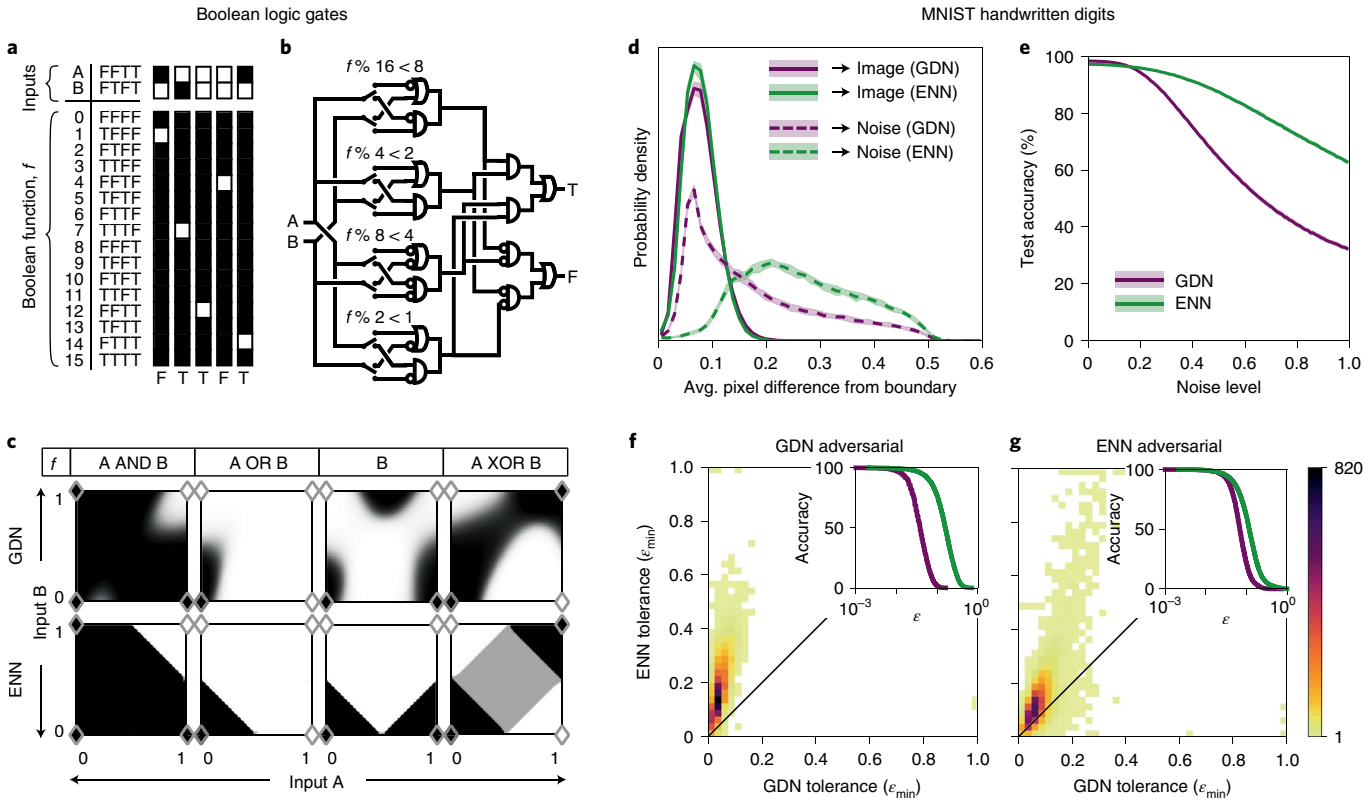

Fig. 5 | Decision boundary robustness to noise and adversarial attacks. a, The truth table for all 16 functions, with five examples of sample encodings. **b**, The ENN-derived logic circuit is possible because of ENN explainability. **c**, Visualized decision boundaries of four logic functions. A and B are inputs, and in grayscale is each networks' output probability for TRUE. Corner diamonds correspond to the training data, with TRUE in white and FALSE in black. **d**–**g**, The results for a GDN and ENN trained on MNIST, with results on the test set. **d**, The probability distributions of the average pixel difference between images and a decision boundary, found by interpolating toward either other test images or white noise (with the interdecile range shaded). **e**, The network classification accuracy as Gaussian noise is added to images (with the interdecile range shaded). **f**,**g**, Adversarial attacks were generated against both the GDN (**f**) and ENN (**g**) for all test images, with the minimum tolerated ϵ_{min} scaling factors shown as two heatmaps (black diagonal identity line for reference). Inset: the classification accuracy at different ϵ values.

Our model was developed to establish principles that are sufficient (though not exclusive) to explain the neural basis of cognition. In doing so, ENNs naturally display emergent properties similar to the brain, such as sparse and modular connectivity, localist firing and short-term neuroplasticity. ENNs should also be well suited to model other neural phenomena. For example, efficient synaptic response to a dynamic environment requires the hierarchical structure and modularity seen in ENNs, as Hebbian learning increases connections between similarly firing neurons[57]. Additionally, recurrent connectivity allows higher-level distinctions to help refine, ignore or emphasize lower-level distinctions, analogous to how deliberative ENNs use recurrent reasoning. Finally, explainable neuron activity would seem to be necessary for introspective metacognition. In Supplementary Table 1, we further suggest how several well-studied neural circuits, and even non-neural biological networks, may relate to the ENN model.

We have kept the ENN structure and learning algorithm simple to demonstrate the power of our simple neurocognitive model to simulate various cognitive capabilities. This means there is substantial potential for improvement. As with gradient-based deep learning, applying ENNs to more complex tasks will require further methodological improvements, such as code optimization, increased network size, improved convolutional filters and data augmentation. Gradient-based methods currently enjoy an advantage in flexibility for various training purposes (for example, latent variables in autoencoders, output neurons in generative networks, and end-to-end training of combined models), which means GDNs will remain powerful tools until future ENN developments can close these gaps. However, ENN explainability and reasoning make possible fundamentally different types of capability, such as deliberation, discovering human-understandable original algorithms, and generalizing out-of-distribution from small training sets. Other possibilities that ENNs should open include incorporating prior human knowledge, designing network architectures and hyperparameters rationally, and learning by definition. Taken together, this work demonstrates the basic principles sufficient for symbolic cognition in neural networks and how this can be simulated in AI systems to overcome current limitations and expand their capabilities.

Methods

General ENN training algorithm. *Terminology.* Each neuron n receives p inputs from other neurons $\mathrm{x}^{(n)}$ and returns the output $y^{(n)} = f\left(\mathbf{w}^{(n)} \cdot \mathbf{x}^{(n)} + b^{(n)}\right)$, where $\mathrm{w}^{(n)}$ are synaptic weights and $b^{(n)}$ is the bias factor. We used the sigmoid activation function $f(x) = \frac{1}{1+e^{-x}}$ for interneurons. For each neuron, the hyperplane $\mathrm{w}^{(n)} \cdot \mathrm{x}^{(n)} + b^{(n)} = 0$ represents its decision boundary. Its positive half-space comprises the points $\left\{\mathbf{x}^{(n)} \in \mathbb{R}^p \mid \mathbf{w}^{(n)} \cdot \mathbf{x}^{(n)} + b^{(n)} > 0\right\}$ and its negative half-space $\left\{\mathbf{x}^{(n)} \in \mathbb{R}^p \mid \mathbf{w}^{(n)} \cdot \mathbf{x}^{(n)} + b^{(n)} < 0\right\}$.

Training sets contain N samples $\mathbf{s}_1, ..., \mathbf{s}_N$, each with target class $c_1, ..., c_N$, where c_i is one of M possible classes, $c_i \in 1, ..., M$. A 'concept' refers to a region of input space for which the neural network predicts a particular class, which must be learned from the set C_i of training samples $\{\mathbf{s}_k | c_k = i, k = 1, ..., N\}$. A 'subconcept' refers to a region in sample space learned from some set of training samples and is usually a subregion of a particular concept, learned from the set of training samples $S_j^{(C_i)} \subseteq C_i$. In the description of the training algorithm below, the terms

concept and subconcept will be used to refer to either the input region or to the corresponding set C or S of training samples interchangeably, which will be clear in context. '(Sub)concept' refers ambiguously to both subconcepts and concept.

Learning subconcepts. ENN construction can be an online-learning or a batch-learning process, and there are several ways to learn subconcepts. In this Article, ENNs were trained with batch learning, and subconcepts were generally learned in an unsupervised manner via hierarchical clustering of the samples in each concept C_i (using the ward clustering metric). In general, the samples may be passed through some transformation g before clustering (that is, $\{g(\mathbf{s})|\mathbf{s}\in C_i\}$). As is standard, the hierarchical trees were cut at a particular level to yield m_i terminal nodes, which represent clusters of training samples forming subconcepts $S_j^{(C_i)}$, $j=1, ..., m_i$, where $\bigcup_{j=1}^{m_i} S_j^{(C_i)} = C_i$.

Learning differentia neuron parameters. Each differentia neuron δ is designed to distinguish between a pair of (sub)concepts $\left(S_j^{(C_i)}, S_l^{(C_k)}\right)$. The differentia neuron receives inputs $g_\delta(\mathbf{s})$. If these inputs come from the network's original input layer, $g_\delta(\mathbf{s})=\mathbf{s}$. In convolutional ENNs, $g_\delta(\mathbf{s})$ are the outputs of the final convolutional layer. The differentia neuron's parameters $\mathbf{w}^{(\delta)}$ and $b^{(\delta)}$ are learned such that the points $\{g_\delta(\mathbf{s})|\mathbf{s}\in S_j^{(C_i)}\}$ fall in the positive half-space and the other points $\{g_\delta(\mathbf{s})|\mathbf{s}\in S_l^{(C_k)}\}$ fall in the negative half-space as best as possible. Linear SVMs were used to learn these parameters, although logistic regression, perceptrons or other algorithms could alternately be used. The parameters can be scaled by a given factor α (that is, $\mathbf{w}^{(\delta)}\rightarrow\alpha\mathbf{w}^{(\delta)}$ and $b^{(\delta)}\rightarrow\alpha b^{(\delta)}$), to effectively change the steepness of the activation function.

Learning (sub)concept neuron parameters. Each (sub)concept neuron σ is designed similarly to differentia neurons, but, instead, they separate a given subconcept $\{g_\sigma(\mathbf{s})|\mathbf{s}\in S_j^{(C_i)}\}$ or concept $\{g_\sigma(\mathbf{s})|\mathbf{s}\in C_i\}$ from all complementary concepts $\{g_\sigma(\mathbf{s})|\mathbf{s}\notin C_i\}$. This means that, although differentia neurons make a one-versus-one distinction between two specific (sub)concepts, (sub)concept neurons make one-versus-all distinctions for individual (sub)concepts (Fig. 1a).

Pruning differentia neurons. It is possible to prune away redundant or less useful differentia neurons from the network to maintain reasonable network sizes. Downstream (sub)concept neurons are trained, and the differentia neurons that they find unnecessary are pruned away. To determine which differentiae were not necessary, we tentatively trained the layer following the differentia layer. For each downstream neuron, the elements in $\mathbf{w}^{(n)}$ with the lowest absolute values signify which differentiae the neuron found least necessary. The neuron was retrained without weights from these unnecessary differentia neurons. This continued iteratively until halting criteria were met, either when the hyperplane misclassified points at a given error rate or when the separation margin (computed in SVM training) decreased by a given percentage (both were hyperparameters). The differentia neurons found unnecessary by all downstream (sub)concept neurons are pruned, and the downstream neurons can then be trained again using only the remaining differentiae.

Basic ENN training algorithm. The training of the basic ENNs consisted of five steps: (1) learn subconcepts within each concept, (2) train differentiae between each pair of subconcepts, (3) prune unnecessary differentiae, (4) train subconcept neurons and (5) train concept neurons.

(1) Hierarchical clustering was performed on training samples from each concept separately. The resulting trees were all cut at the same value, which was determined such that there was a given number of total subconcepts (that is, $\sum_{i=1}^{M} m_i$), specified by a hyperparameter.

(2) Differentia neurons were designed for all pairs of unrelated subconcepts $(S_j^{(C_i)}, S_l^{(C_k)})$ for $i,k=1, ..., M$, $i\neq k$, with $j=1, ..., m_i$, and $l=1, ..., m_k$.

(3) Differentia neurons were optionally pruned by tentatively training the subconcept layer neurons.

(4) Subconcept neurons were trained using outputs from the remaining differentia neurons, separating $S_j^{(C_i)}$ from $\bigcup_{k\neq i} C_k$ for all $i=1, ..., M$ and $j=1, ..., m_i$.

(5) Concept neurons were trained using outputs from the subconcept neurons. These neurons generally used the softmax activation function. Concept neuron weights were allowed to be improved with gradient descent, although generally the change was small and effectively just scaled all the concept neuron weights proportionally.

ENN hyperparameters. There are several hyperparameters used for training basic ENNs: the number of subconcepts; an SVM cost to set the softness of the margin; SVM scaling factors α for each layer; the differentia pruning halting conditions (maximum margin decrease and/or maximum misclassification tolerance). To find optimal hyperparameters, grid or random search was done using 10-fold cross-validation. To speed up the search, we performed this on subsets of the training data. We often chose multiple final values for the number of subconcepts and for the maximum margin decrease so as to vary the size of the network, such that the network size would be generally comparable to previously published work[5,58]. For example, the hyperparameters of a basic ENN trained on MNIST (Table 1) comprised 60 subconcepts, SVM costs of 15 and 1,000 for the differentia and subconcept layers, SVM scaling factors of 8 for differentia and subconcept layers and 4 for the concept layer, and pruning halting conditions of 96% maximum margin decrease.

Other ENN architectures. *Convolutional ENNs.* Subimages from the training set were randomly sampled uniformly across locations in the images, equally from each class, and k-means clustering was used to learn 'feature subconcepts', with k corresponding to the number of convolutional filters (a hyperparameter). Each cluster was collapsed to its mean, and subconcept one-versus-all SVMs were computed for each convolutional filter. The outputs of the convolutional layer were passed through a max-pooling layer. These outputs were used to generate another set of convolutional filters and a max-pooling layer. The outputs of the second max-pooling layer were then fed into the basic ENN training algorithm, using the final convolutional layer output as inputs. The hyperparameters in convolutional ENNs included the number and size of filters, stride rates, an SVM scaling factor and a max-pooling size.

The MNIST input images were padded out to be 32×32 pixels, consistent with LeNet-5[5]. The smaller of the two convolutional ENNs in Table 1 was designed to be of similar dimensions to LeNet-5. Filters were visualized both by plotting weights and by computing the weighted average of all windows in the test set that lie on either side of the filter's hyperplane (for a fair comparison between ENNs and GDNs). The weight applied to each window with filter output y_i was $|y_i-0.5|$.

Deliberative ENNs. In deliberative ENNs, deliberation occurs whenever the maximum output probability for a test sample is less than 0.5. The network uniformly modulates the bias of its subconcept neurons by adding or subtracting a small constant, stopping once some concept neuron's output exceeds 0.5. Biases are positively modulated when no subconcept neurons fire above 0.5 and are negatively modulated otherwise.

Hierarchical ENNs. The hierarchical linkage tree for each concept found in step (1) of the basic ENN training algorithm is traversed backwards to include clusters represented by nodes at different levels in the tree. Within a concept, these subconcepts include the usual basic ENN subconcepts as well as additional subconcepts that are supersets of basic subconcepts. Each training sample is therefore assigned to multiple subconcepts at various depths in the hierarchical tree. All the basic ENN subconcept neurons were in a single layer after the differentia neurons, and then each additional subconcept found traversing the tree was represented by a subconcept neuron in subsequent layers. Deeper subconcept layers contained at most one subconcept neuron per concept. Each subconcept neuron received connections from all differentia neurons as well as all subconcept neurons from previous layers, giving rise to skip connections between non-adjacent layers (equivalently, $g_\sigma(\mathbf{s})$ for each subconcept neuron σ contained the outputs from the differentia layer and previous subconcept layers). The output concept neurons only received input from subconcept neurons.

Consensus ENNs. We trained consensus ENNs by learning two sets of subconcepts. In addition to basic ENN subconcepts learned within concepts, we also performed hierarchical clustering on the full set of training samples unsupervised, cutting the linkage tree to form a given number of new subconcepts S'_j. These new subconcepts could overlap multiple concepts, with $\bigcup_j S'_j = \bigcup_i S_i$. Training otherwise proceeded as with basic ENNs.

Symbolic ENNs. The sigmoid activation function was replaced in basic ENNs with the Heaviside step function:

$$f(x)=\begin{cases}0 & \text{if } x<0\\ 0.5 & \text{if } x=0\\ 1 & \text{if } x>0\end{cases}$$

GDNs. Each GDN was trained using the same architecture as its comparison ENN. They were each trained using Keras with the Adam optimizer with default parameters and a categorical cross-entropy loss function. Tenfold cross-validation was used to find the optimal batch size and number of training epochs for all GDNs except the GDNs computed for Supplementary Fig. 1, which instead used fivefold cross-validation.

Datasets. *Image datasets.* The rectangles dataset was synthetically generated, similar to ref. [58]. Each 28×28 black image had a white filled rectangle oriented horizontally or vertically. There were 50,000 training images and 10,000 test images.

The MNIST dataset consists of 60,000 training images and 10,000 test images, each of which is a 28×28 grayscale image with a single handwritten digit 0 through 9[5]. The CIFAR-10 dataset consists of 50,000 training images and 10,000

test images, each of which is a 32×32-RGB image from one of 10 classes of animals or vehicles[45].

For the orientation problem, the 56 training images were 28×28 black images with a one-pixel-wide stripe across the full length or height of the image. To generate the diagonal line and rectangle outline target sets, for every combination of possible heights and widths of non-square rectangles we made at most 50 images with random translations of the rectangle or one of its diagonals. The dotted line target set took the training images and turned them into dotted lines of varying intervals (spacing of 1–26 pixels). The zigzag dataset took two horizontally or vertically oriented endpoints and drew a zigzag between them with each line segment at 45° (for total zigzag widths 3–13).

Boolean logic problem. The full two-input truth table for all 16 Boolean functions, as displayed in Fig. 5a, was encoded into 64 individual samples. Each sample contained 18 features, the first two encoding the function inputs (as either 1 or −1) and the other 16 one-hot encoding the function index.

The logic circuit in Fig. 5b uses four pairs of switches controlled by the indicated inequality, where f is the index of the Boolean function. The plots in Fig. 5c were generated by testing each network on a fixed function index for different interpolated function inputs.

MAX-SAT. Formulae in conjunctive normal form with m clauses and n possible variables were encoded in $m \times 2n$ matrices, with each row corresponding to a clause and each column corresponding to some variable x_i or (not x_i). Training samples were generated by choosing a single clause and including either (x_1) or (not x_1). Samples could also have one additional variable (x_i) or (not x_i). All such possible formulae were included in the training sets, for a total of $m(4n-2)$. The ideal x_1 assignment was encoded in two values corresponding to TRUE and FALSE: (1, 0) for TRUE, (0, 1) for FALSE, (0.99, 0.01) for TRUE/FALSE and (0.01, 0.99) for FALSE/TRUE. The latter two were used to indicate the network should prefer one assignment over the other, but that the other was still possible.

To generate the two test sets, 5,000 unique formulae were generated, each with a random number of empty clauses. For MAX-3SAT, each clause contained three randomly chosen variables. For MAX-SAT, each variable was included with probability $3/n$.

Each algorithm (ENN, GDN, random, pure greedy or 3/4-approximation) returns a probability of assigning TRUE or FALSE to x_1. After randomly choosing based on this probability, all newly satisfied clauses were emptied. The variable indices were then shifted over one so that the algorithms could then choose an assignment for each subsequent variable. The results were averaged over 100 trials because of the probabilistic assignments and because the 3/4-approximation algorithm's guarantee is for the expected number of satisfied clauses. Accuracy is reported as the difference with respect to the pure greedy algorithm.

The TSP. Maps were unit squares containing 10 cities to be visited. The network inputs included the upper half of the inter-city distance matrix (45 values) and a one-hot encoding indicating the salesman's current city. Already-visited cities were denoted as being a distance of 10 from all other cities. The training set consisted of 90 maps, each with only one unvisited city, with the correct next city located a very short distance from the current city. The test set had 5,000 maps with 10 randomly located cities.

ENNs were deliberative and had each subconcept only connected to its associated differentiae, and the output neurons used the sigmoid activation function. After training, each network was asked to find a route for the test set maps. After the network picked a city to move to, the distance matrix was updated by setting all distances to and from the previous city to 10 and changing the one-hot encoding to the new city. The network outputs corresponding to cities already visited were masked to prevent the possibility of endless loops. The nearest-neighbor greedy algorithm (which chooses the closest unvisited city to visit next) served as a reference for network performance, with the reported test accuracy being the average difference in route lengths found by the network and the reference.

The BDT problem. The problem is to find a BDT that reproduces a given truth table with minimum tree depth, defined as the average depth necessary to classify each entry of the truth table. Samples came from 10-input truth tables that contained 1,024 entries, with the label column serving as the network input (encoded as zeros and ones). The target output of the network is which of the 10 inputs to split at that node of the tree. The training set consisted of just 20 samples corresponding to the truth tables associated with all possible BDTs containing a single branch node. For the test set, we generated 5,000 random BDTs by allowing each node to branch with probability 0.7 and not allowing branches beyond depth 7, with the leaf labels assigned randomly. The truth tables of these BDTs formed the test set.

To build a BDT with a trained network, the network chose which input to split for the first branch node. Going down each of the branches in turn, if all entries on the branch were labeled the same, a leaf was placed at the end with the corresponding label. If more branch nodes were necessary, the truth table was temporarily reformed by taking the truth table half corresponding to its side of the split and copying onto the other half, such that the already-split features no longer needed to be split. This new truth table was put through the network again, masking the outputs of already chosen inputs to prevent infinite trees. This was done recursively until all branches terminated in leaves.

Each test sample was also given to the greedy CART algorithm with Gini impurity as the splitting criterion, using scikit-learn's DecisionTreeClassifier. The test accuracy reported is the average difference in tree depth found by a neural network compared to the CART algorithm.

Performance metrics. Networks were trained and tested on a single central processing unit node on the BioHPC computing cluster at the University of Texas Southwestern Medical Center. Training times are the measured wall times, starting after loading the training data and ending after the network's parameters were finalized. The reported accuracies come from the test sets, which were excluded from training and hyperparameter optimization.

Neuron functional groups. For modularity analysis and visualization, each neuron was duplicated and the outgoing weights $\mathbf{w}^{(n)}$ became either $\min(0, \mathbf{w}^{(n)})$ or $\max(0, \mathbf{w}^{(n)})$. Each neuron's output was artificially altered between 0 and 1 for 5,000 test images, and we found the output neuron's activity that changed most. For each connectivity matrix, we computed the Kolmogorov–Smirnov statistic between the within-group weight distribution (that is, weights between neurons assigned to the same group) and the between-group weight distribution (that is, weights between neurons assigned to different groups). To assess functional importance, we drove each neuron's output between 1 and 100 and measured the maximum change in the output layer. Supplementary Fig. 4b plots the average response for each neuron.

Explaining errors. Each differentia neuron's output was artificially varied to see if it could individually change the network output to be correct for a misclassified test image. The misleading features figures were generated by calculating $((D \odot I)_+ - (D \odot S)_+)_+$, where I is the test image, D is the differentia weight set, S is the average of all training images from the correct subconcept, and $(x)_+ = \max(x, 0)$. The missing features figures were generated by calculated $(((-D) \odot S)_+ - ((-D) \odot I)_+)_+$. These assumed that the correct subconcept is in the differentia neuron's positive half-space, otherwise the sign of D is reversed.

Translating symbolic ENNs to pseudocode. Symbolic ENN weights were examined and translated into pseudocode that performed the equivalent step-by-step algorithm. This was done for each neuron, determining what its weights did in the space of its inputs. This, usually, was some form of threshold logic that we have written either as if-statements with inequality conditions, or sometimes as simple logic functions like OR and AND.

Choosing optimal GDN network sizes for generalization. For the orientation problem and the TSP, we trained GDNs of varying layer widths, performing a grid search by scaling from 0 to twice the width of each ENN layer. We trained ten GDNs for the orientation problem and five for the TSP. We then chose the architecture with the best average performance on the test sets as the best-architecture GDN.

Seeding GDNs with ENN weights. To demonstrate the rarity of finding a generalizable solution with GDNs, we took the symbolic ENN, scaled its weights by 16 and changed the activation functions to sigmoids, then perturbed its weights by a small amount before training as a GDN. The perturbation consisted of adding a normally distributed value to all weights and biases, with the standard deviation being a given fraction of the mean weight magnitude for each layer separately.

Network lesions. Lesions were performed in the second hidden layer (subconcept neurons in ENNs). Neurons were deleted sequentially, and test accuracy was calculated individually for each class. The sequence of neuron deletions was determined by hierarchical linkage clustering on the neurons' outputs on the test set, assuming that neurons with similar firing patterns are physically located more closely.

Distances to decision boundaries. For each sample in the test set correctly predicted by both the ENN and GDN—about 96% of MNIST and 99% of the rectangles test sets—20 target locations were chosen for interpolation. Each target was either a test image from a different class or white noise (that is, random black and white pixels) that the networks classified differently than the test image. Interpolating between the sample and the target, the point at which the network changed its predicted class was found and the average pixel difference was calculated (which is proportional to the L_1 distance to the boundary).

Robustness analyses. To test robustness to noise, Gaussian noise with various standard deviations was added to test images, repeating 20 times for each noise level.

To generate adversarial images, we used the fast gradient sign method, where perturbations are given by $\epsilon \mathrm{sign}(\nabla_{\mathbf{x}} L)$, where $\mathbf{x}$ is the original image and L is the network's loss function[8]. We increased the value of ϵ until the image

was misclassified ($\epsilon_{\min}$). For both the GDN and ENN, we generated adversarial perturbations for each test image, where L for both was the cross-categorical entropy loss used to train the GDN. $\epsilon_{\min}$ was found separately for each network.

Data availability

The datasets used in this work are included with the code[59]. The MNIST and CIFAR-10 datasets are publicly available[5,45]. Source data are provided with this paper.

Code availability

The code used to build, train and analyze ENNs as well as the various training and test sets have been deposited in Code Ocean[59].

Received: 22 December 2020; Accepted: 16 August 2021;
Published online: 22 September 2021

References

1. Minsky, M. Logical vs. analogical or symbolic vs. connectionist or neat vs. scruffy. *AI Magazine* **12**, 34–51 (1991).
2. Gardenfors, P. Conceptual spaces as a framework for knowledge representation. *Mind Matter* **2**, 9–27 (2004).
3. Lecun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
4. McClelland, J. L. & Rogers, T. T. The parallel distributed processing approach to semantic cognition. *Nat. Rev. Neurosci.* **4**, 310–322 (2003).
5. Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998).
6. Krizhevsky, A., Sutskever, I. & Hinton, G. E. ImageNet classification with deep convolutional neural networks. In *Proc. Advances in Neural Information Processing Systems 25* 1097–1105 (NIPS, 2012).
7. Shwartz-Ziv, R. & Tishby, N. Opening the black box of deep neural networks via information. Preprint at https://arxiv.org/pdf/1703.00810.pdf (2017).
8. Szegedy, C. et al. Intriguing properties of neural networks. Preprint at https://arxiv.org/pdf/1312.6199.pdf (2013).
9. Marcus, G. Deep learning: a critical appraisal. Preprint at https://arxiv.org/pdf/1801.00631.pdf (2018).
10. Kindermans, P.-J. et al. in *The (Un)reliability of Saliency Methods* 267–280 (Springer, 2019).
11. Olah, C. et al. The building blocks of interpretability. *Distill* **3**, e10 (2018).
12. Lake, B. M., Salakhutdinov, R. & Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science* **350**, 1332–1338 (2015).
13. Wang, M. & Deng, W. Deep visual domain adaptation: A survey. *Neurocomputing* **312**, 135–153 (2018).
14. McCulloch, W. S. & Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* **5**, 115–133 (1943).
15. Han, S., Pool, J., Tran, J. & Dally, W. Learning both weights and connections for efficient neural network. In *Proc. Neural Information Processing Systems 28* 1135–1143 (NIPS, 2015)
16. Happel, B. L. & Murre, J. M. Design and evolution of modular neural network architectures. *Neural Netw.* **7**, 985–1004 (1994).
17. Roy, A. A theory of the brain: localist representation is used widely in the brain. *Front. Psychol.* **3**, 551 (2012).
18. Quiroga, R. Q. & Kreiman, G. Measuring sparseness in the brain: comment on Bowers (2009). *Psychol. Rev.* **117**, 291–297 (2010).
19. Roy, A. A theory of the brain—the brain uses both distributed and localist (symbolic) representation. In *Proc. 2011 International Joint Conference on Neural Networks* 215–221 (2011).
20. Tolhurst, D. J., Smyth, D. & Thompson, I. D. The sparseness of neuronal responses in ferret primary visual cortex. *J. Neurosci.* **29**, 2355–2370 (2009).
21. Yen, S.-C., Baker, J. & Gray, C. M. Heterogeneity in the responses of adjacent neurons to natural stimuli in cat striate cortex. *J. Neurophysiol.* **97**, 1326–1341 (2007).
22. Richards, B. A. & Lillicrap, T. P. Dendritic solutions to the credit assignment problem. *Curr. Opin. Neurobiol.* **54**, 28–36 (2019).
23. Bengio, Y., Lee, D.-H., Bornschein, J., Mesnard, T. & Lin, Z. Towards biologically plausible deep learning. Preprint at https://arxiv.org/pdf/1502.04156.pdf (2016).
24. Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T. & Maida, A. Deep learning in spiking neural networks. *Neural Netw.* **111**, 47–63 (2019).
25. Wittgenstein, L. *Philosophical Investigations* 3rd edn (Basil Blackwell, 1968).
26. Maddox, W. & Ashby, F. Comparing decision bound and exemplar models of categorization. *Percept. Psychophys.* **53**, 49–70 (1993).
27. Ashby, F. & Maddox, W. Human category learning. *Annu. Rev. Psychol.* **56**, 149–178 (2005).
28. Aristotle. Book III. In *De Anima* (ed. Ross, W. D.) (Oxford: Clarendon Press, 1931).
29. Aquinas, T. Prima Pars. In *Summa Theologiae*. q78 (Ave Maria Press, 2000).
30. Aquinas, T. Question 1: Truth. In *Quaestiones disputatae de Veritate* (ed. Mulligan, R.) (Henry Regnery Company, 1952).
31. Hume, D. Book 1: Of the Understanding. In *A Treatise of Human Nature* (ed. Selby-Bigge, L.A.) (Oxford: Clarendon Press, 1896).
32. Kant, I. Introduction. In *Critique of Pure Reason* (eds. Guyer, P. & Wood, A.) (Cambridge Univ. Press, 1998).
33. Aristotle. Book VII. In *Metaphysics* (ed. Ross, W. D.) (Oxford: Clarendon Press, 1924).
34. Aristotle. Section I. In *Categories* (ed. Ross, W. D.) (Oxford: Clarendon Press, 1928).
35. Bonaventure. Chapter III. In *Itinerarium Mentis in Deum* (ed. Cousins E.) (Paulist Press, 1978).
36. Aquinas, T. *De Ente et Essentia* (ed. Bobik, J.) (University of Notre Dame Press, 1965).
37. Aristotle. *Posterior Analytics* (ed. Ross, W. D.) (Oxford: Clarendon Press, 1925).
38. Aquinas, T. Prima Pars. In *Summa Theologiae*. q79 (Ave Maria Press, 2000).
39. Hobhouse, L. T. *The Theory of Knowledge: a Contribution to Some Problems of Logic and Metaphysics* 3rd edn (Methuen & Co., 1921).
40. Gurney, K. in *An Introduction to Neural Networks* Ch. 3 (Taylor & Francis, 1997).
41. Bellmund, J. L. S., Gardenfors, P., Moser, E. I. & Doeller, C. F. Navigating cognition: spatial codes for human thinking. *Science* **362**, eaat6766 (2018).
42. Reich, D. S., Lucchinetti, C. F. & Calabresi, P. A. Multiple sclerosis. *N. Engl. J. Med.* **378**, 169–180 (2018).
43. Korczyn, A. D., Vakhapova, V. & Grinberg, L. T. Vascular dementia. *J. Neurol. Sci.* **322**, 2–10 (2012).
44. Evans, J. S. B. T. Dual-processing accounts of reasoning, judgment and social cognition. *Annu. Rev. Psychol.* **59**, 255–278 (2008).
45. Krizhevsky, A. *Learning Multiple Layers of Features from Tiny Images*, Technical Report TR-2009 (Univ. Toronto, 2012).
46. Jha, D. et al. Lightlayers: parameter efficient dense and convolutional layers for image classification. In *Parallel and Distributed Computing: Applications and Technologies* (eds Zhang, Y., Xu, Y. & Tian, H.) 285–296 (Springer, 2021).
47. Brette, R. Philosophy of the spike: rate-based vs. spike-based theories of the brain. *Front. Syst. Neurosci.* **9**, 151 (2015).
48. Qiao, F., Zhao, L. & Peng, X. Learning to learn single domain generalization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 12553–12562 (IEEE Computer Society, 2020).
49. Arnold, A., Nallapati, R. & Cohen, W. W. A comparative study of methods for transductive transfer learning. In *Proc. Seventh IEEE International Conference on Data Mining Workshops* (*ICDMW 2007*) 77–82 (IEEE, 2007).
50. Hu, S., Zhang, K., Chen, Z. & Chan, L. Domain generalization via multidomain discriminant analysis. In *Proc. Machine Learning Research* Vol. 115 (eds Adams, R. & Gogate, V.) 292–302 (PMLR, 2020).
51. Karp, R. Reducibility among combinatorial problems. *Complexity Comput. Comput.* **40**, 85–103 (1972).
52. Johnson, D. S. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.* **9**, 256–278 (1974).
53. Poloczek, M., Schnitger, G., Williamson, D. & Zuylen, A. Greedy algorithms for the maximum satisfiability problem: simple algorithms and inapproximability bounds. *SIAM J. Comput.* **46**, 1029–1061 (2017).
54. Hyafil, L. & Rivest, R. L. Constructing optimal binary decision trees is NP-complete. *Inf. Process. Lett.* **5**, 15–17 (1976).
55. Bose, N. K. & Garga, A. K. Neural network design using Voronoi diagrams. *IEEE Trans. Neural Netw.* **4**, 778–787 (1993).
56. Riesenhuber, M. & Poggio, T. Hierarchical models of object recognition in cortex. *Nat. Neurosci.* **2**, 1019–1025 (1999).
57. Cooper, L. N. & Bear, M. F. The BCM theory of synapse modification at 30: interaction of theory with experiment. *Nat. Rev. Neurosci.* **13**, 798–810 (2012).
58. Bergstra, J. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **13**, 281–305 (2012).
59. Blazek, P. J. & Lin, M. M. Essence neural networks. *CodeOcean* https://doi.org/10.24433/CO.7389497.v1 (2021).

Acknowledgements

We acknowledge the Cecil H. and Ida Green Foundation, the Welch Foundation (grant no. I-1958-20180324) and the anonymous-donor-supported UTSW High Risk/High Impact grant for funding this research.

Author contributions

P.J.B. and M.M.L. designed the research. P.J.B. performed the research, contributed new analytical tools and analyzed data. P.J.B. and M.M.L. wrote the manuscript.

Competing interests

The authors have filed an international patent related to this work (PCT/US2021/019470).

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s43588-021-00132-w.

Correspondence and requests for materials should be addressed to Milo M. Lin.

Peer review information *Nature Computational Science* thanks the anonymous reviewers for their contribution to the peer review of this work. Handling editor: Ananya Rastogi, in collaboration with the *Nature Computational Science* team.