Neurocompositional computing in human and machine intelligence: A tutorial

Paul Smolensky,[1,2*] R. Thomas McCoy,[1*] Roland Fernandez,[2]
Matthew Goldrick,[3] Jianfeng Gao[2]

[1]Department of Cognitive Science, Johns Hopkins University
3400 N. Charles St., Baltimore, MD 21218, USA
[2]Microsoft Research; Redmond, WA 98052, USA.
[3]Department of Linguistics, Northwestern University; Evanston, IL 60208, USA.

[*]To whom correspondence should be addressed;
E-mail: {tom.mccoy,paul.smolensky}@jhu.edu.

Wednesday 4th May, 2022

Contents

Abstract

The past decade has produced a revolution in Artificial Intelligence (AI), after a half-century of AI repeatedly failing to meet expectations. What explains the dramatic change from 20th-century to 21st-century AI, and how can remaining limitations of current AI be overcome?

Until now, the widely accepted narrative has attributed the recent progress in AI to technical engineering advances that have yielded massive increases in the quantity of computational resources and training data available to support statistical learning in deep artificial neural networks. Although these quantitative engineering innovations are important, here we show that the latest advances in AI are not solely due to *quantitative* increases in computing power but also *qualitative* changes in how that computing power is deployed. These qualitative changes have brought about a new type of computing that we call **neurocompositional computing**.

In neurocompositional computing, neural networks exploit two scientific principles that contemporary theory in cognitive science maintains are simultaneously necessary to enable human-level cognition. The Compositionality Principle asserts that encodings of complex information are structures that are systematically composed from simpler structured encodings. The Continuity Principle states that the encoding and processing of information is formalized with real numbers that vary continuously. These principles have seemed irreconcilable until the recent mathematical discovery that compositionality can be realized not only through the traditional discrete methods of symbolic computing, well developed in 20th-century AI, but also through novel forms of continuous neural computing—neurocompositional computing.

The unprecedented progress of 21st-century AI has resulted from the use of limited—first-generation—forms of neurocompositional computing. We show that the new techniques now being deployed in second-generation neurocompositional computing create AI systems that are not only more robust and accurate than current systems, but also more comprehensible—making it possible to diagnose errors in, and exert human control over, artificial neural networks through interpretation of their internal states and direct intervention upon those states.

Note: This tutorial is intended for those new to this topic, and does not assume familiarity with cognitive science, AI, or deep learning. Appendices provide more advanced material. Each figure, and the associated box explaining it, provides an exposition, illustration, or further details of a main point of the paper; in order to make these figures relatively self-contained, it has sometimes been necessary to repeat some material from the text. For a brief introduction and additional development of some of this material see [212].

Overview

A landmark achievement of the 21st century is the widespread deployment of Artificial Intelligence (AI) [117, 220], a turning point that has been compared to the industrial revolution of the 18th century [224]. Properly understanding this breakthrough is critical for controlling how AI develops in the future, both for maximizing its benefits and minimizing its hazards [62]. Adopting a cognitive-science perspective, we present a new analysis of this progress, an analysis which has led to a new generation of AI systems that we introduce.

0 Neurocompositional computing

AI has been a long time in the making. It was in 1843 that the real possibility of AI arose through a confluence of philosophy and technological design [65]. For millennia, philosophers had argued that rational thought was a type of calculation over ideas [49, 195]. Then, in 1838, Charles Babbage conceived a mechanical device for general calculation over numbers [18]. It was Ada Lovelace who quickly recognized the profound implications, extending far beyond numerical calculation: with the machine, "in enabling mechanism to combine together *general* symbols, in successions of unlimited variety and extent, a uniting link is established between the operations of matter and ... abstract mental processes" [14, Note A, p. 368].

A century later, engineering caught up with theory. In the 1940s, general-purpose computing machines were physically realized [178], and the power of such machines as a bridge to mental processes was soon manifest in the birth of AI [100, 215]. Although Lovelace herself was skeptical that computers could ever generate original ideas [14, Note G, p. 398], eminent computer scientists since the 1950s have predicted that machines would match human intelligence within a few decades: by 2000 (Alan Turing, writing in 1950 [228, p. 442]); by 1973 (John McCarthy, writing in 1963 [149, p. 20]); by 1985 (Herbert Simon, writing in 1965 [194, p. 96]); or by about 1992 (Marvin Minsky, writing in 1967 [144, p. 2]) (all cited in [40, p. 109]).

Such predictions pervaded not just science but also science fiction: futurist Arthur C. Clarke's 1968 work *2001: A Space Odyssey* features a computer named HAL-9000 capable of sophisticated conversation and planning [36]. As each predicted arrival date came and went, however, the technology consistently fell short of the predictions [143] (see [40, Ch. 5] for a review). Even today—over two decades after the titular year for *2001: A Space Odyssey*—contemporary AI systems still cannot come close to HAL-9000's sophisticated conversational abilities (not to mention its skill in planning a mutiny) [19].

But while they are still a far cry from human-level cognition, AI systems in the new millennium have now achieved feats unimaginable even 15 years ago. AI systems now power everyday technology from search engines [180] to smartphone apps [101], and have decisively defeated human champions in games ranging from the quiz show Jeopardy! [24] to the famously intractable board game Go [192].[1] Perhaps most impressively, AI has enabled breakthroughs in longstanding scientific grand challenge problems such as predicting protein folding [4, 189] and modeling turbulent fluid flow [119].

This recent progress in AI—specifically, AI based on artificial neural networks exploiting deep learning [12, 110] (§1.2.4, p. 22)—has typically been attributed to merely quantitative technological advances: greatly increased computing power and data quantity [109]. Correct as far as it goes, this view overlooks a major *qualitative* factor, namely the advent of a new type of computing that we call **neurocompositional computing** [212]: the form of computing underlying human intelligence, according to a contemporary theory in cognitive science [210]. We argue that, in addition to the critical expansion of computing resources,[2] it is the unrecognized emergence of neurocompositional computing—albeit

[1] Neural-computing-based AI may even be destined to become an everyday part of continuous, real-time strategizing in American football [238].

[2] Vital computing resources include massive datasets for learning the statistical patterns hidden within AI

only implicitly, in an acutely restricted form—that has powered the dramatic progress in AI in the 21st century.

In this tutorial we present neurocompositional computing and show how it has enabled progress on three fundamental problems that have challenged current-generation AI (1).

(1) **Fundamental challenges for contemporary AI**

a. compositional generalization

b. data efficiency

c. comprehensibility

The first problem concerns **compositional generalization**, the ability to correctly solve any novel problem that is composed of familiar parts; here current neural AI systems are erratic, falling significantly short of the robust abilities of human cognition [53, 75, 160]. The second is poor **data efficiency**: the partial compositional generalization that neural AI systems do achieve is dependent on orders of magnitude more learning data than is required by human learners [123]. The last is the extremely limited extent to which current AI systems are **comprehensible**, which introduces serious impediments for the use and further development of AI [2, 227].

The framework we present for addressing these three problems derives from a perspective on intelligence that has arisen in cognitive science, a field that emerged in close connection with AI in the 1950s [154]. What fueled the recent unprecedented advances in AI were developments in understanding human cognition, enabled by the fundamental insight that what we have in *our* heads is a computer [172]. To bring the level of general intelligence in machines closer to that of humans, the framework presented here extends the cross-fertilization of AI and cognitive science by *bringing the* ***type of computing*** *used in AI systems closer to that of the human cognitive computer*.

This turns out to require integrating the types of computing used by traditional and contemporary AI, which, as we discuss below, are respectively modeled on computing in the human *mind* and computing in the human *brain*. Recent progress in cognitive science provides a unified mathematical theory of computing in the human *mind/brain*: neurocompositional computing. Emulating this type of computing provides a road map for next-generation AI. This new type of AI exploits the major strengths of each of traditional and contemporary AI to overcome the serious limitations inherent in the other (Fig. 1, p. 9).

The work presented here realizes neurocompositional computing through techniques developed for modeling the human mind/brain: **Neurally-Encoded Compositionally-Structured Tensor (NECST)** computing. We illustrate how this NECST-generation of AI systems yields progress in the three areas of weakness of current AI identified in (1).

For reference, the main contributions of the work presented here are previewed in (2); many of these points are illustrated in the figures cited. A detailed summary is given below in (6), p. 52.

tasks, as well as sheer computing power: the speed of a processor's basic operations, the number of processors that can efficiently compute simultaneously (increased by algorithmic, software and hardware innovations), data storage capacity, higher-level programming languages that automatically perform calculus, and massive internet sharing of program code supporting a large community of researchers.

(2) **Contributions of this work**

We provide evidence for the following claims:

a. Key to the possibility of human intelligence is that cognitive computing simultaneously respects two general principles: the Compositionality Principle—the basis of traditional AI—and the Continuity Principle—the basis of most contemporary AI systems. Together, these define neurocompositional computing (§1, p. 14).

b. Neurocompositionality unifies compositional-structure computing with neural computing, using the strengths of each to overcome major limitations of the other (Fig. 1, p. 9, discussed throughout the tutorial).

c. Neurocompositionality is a 'Structure-in-Neurons' approach to unifying compositional and neural computing, which offers the potential for modeling the general intelligence typical of human cognition; 'Neurons-in-Structure' hybrid computing does not, although it has great power in particular formal problem domains which, unlike most natural domains, are purely discrete by construction (Fig. 2, p. 20).

d. The extremely productive Transformer architecture, responsible for much of the recent jump in AI performance, gains its power from using first-generation (1G) neurocompositional computing (§2.2.2.2, p. 36).

e. Deeper, 2G neurocompositional computing can be achieved through techniques developed in cognitive science including NECST computing, presented informally in the text (§2, p. 24) and more formally in Appendix A (Fig. 9, p. 55); a visualization and analogy are also offered (Fig. 3, p. 25). Theoretical results show that NECST computing affords neural networks key elements of the power of symbolic compositional-structure processing, and advances the theory of natural language grammars (§2, p. 24 and Fig. 11, p. 60).

f. Neurocompositional models use deep learning to invent their own forms of compositional-structure representation and processing (Fig. 5, p. 41; Fig. 6, p. 44).

g. 2G neurocompositional models, in particular numerous NECST-generation models (§3, p. 40), make significant progress against the challenges in (1*a*–*b*) (Fig. 4, p. 37; Fig. 7, p. 46).

h. Addressing challenge (1*c*), the additional comprehensibility of neurocompositional models affords important benefits, including informing structure learning, diagnosing errors, and controlling neural network behavior (Fig. 8, p. 48).

0.1 The computational anatomy of human intelligence: Two principles

To emulate the type of computing used in cognition, AI systems need to match the human computer in two fundamental respects. To process information, a computer needs that information to be encoded in some form that the machine can operate on: the form of these information ***encodings*** is the first fundamental characteristic of a type of computing. The second is the set of basic *operations* the machine can perform on these encodings, and the ways these simple operations can be composed together to create more complex operations.

So, then, what *is* the type of computing used in cognition? It seems clear that the entire machine is a neural network. Information is encoded within the brain by the numerical activation levels of large populations—or 'layers'—of neurons, and is processed by passing this activation through myriad synapses of varying strengths that interconnect these neurons. In this ***neural computing*** [34], both the encoding and the operations performed on them respect the **Continuity Principle**: they are both continuous in that they are formalized with real numbers, which can vary to an arbitrarily small degree. It is mathematical theories of human neural computing that led ultimately to the deep learning revolution in AI.[3]

The neural-network organization of cognition may appear obvious. But since antiquity [99, 195], virtually all aspects of intelligence in the human mind have been understood in terms of a very different type of computing: ***compositional structure processing*** [86]. In this type of computing, complex information is encoded in large structures—*compositional encodings*—which are built by composing together smaller substructures that encode simpler information. These encodings respect the **Compositionality Principle**: to process the complex information encoded in a large structure, it suffices to compose together the results of processing the simpler information encoded in the smaller substructures [54, 147, 222]. The decomposition of information provided by effective compositional encodings carves the task domain at its joints.

To exploit compositionality, a computer must possess compositional encodings as well as the basic operations that compose smaller encodings together to form larger ones, and basic operations that decompose larger encodings into their smaller parts. These requirements are the focus of this tutorial. In addition, the computer must have the means to process the individual parts appropriately to achieve its task: this involves much further analysis that goes beyond the scope of this tutorial (but see, e.g., [128]).

Compositionality has long been seen as a key to the power of human cognition [75]: it provides strong compositional generalization, enabling us to understand any one of a potentially infinite number of novel states of the world by encoding the state internally as a novel composition of familiar, simpler parts, and composing together our understanding

[3] Methods widely used by deep learning practitioners today that were developed by cognitive scientists in the 1980s and 1990s include the back-propagation learning algorithm [182]—the foundation of deep learning—as well as energy methods [79] including Boltzmann Machines, both Restricted [196] and general [1]), and other methods of unsupervised learning [74], reinforcement learning [235], Convolutional Neural Networks [111], next-symbol prediction and symbol-sequence-to-symbol-sequence mapping [51, 90], Long-Short-Term-Memory recurrent networks [76], tree-structured networks for recursive processing [168], neural networks for learning finite-state-machines [44] and grammars [190], mixtures of experts [84], network distillation [150], distributional vector word embeddings [107], the vector-difference model of analogies [181], and many other trailblazing techniques [72, 183].

of those parts (but cf. [161]). In virtually all domains of cognition—from vision and speech to reasoning and planning—extensive empirical investigation over centuries has consistently shown how cognitive functions can be well approximated as computation over appropriate compositional structures, the structures with respect to which human cognition exhibits strong compositional generalization. The parts which compose to form such encodings are sometimes familiar elements like objects, words, concepts, and actions, and sometimes scientifically-discovered units like phonemes—the basic sounds of speech which combine to form words, each of which is typically denoted by a single letter in alphabetic writing systems. Robust compositionality is all-pervasive in cognition: it underlies the power of both fast, automatic, intuitive, largely unconscious cognition (the 'System 1' of [94], from [218]), as well as slow, controlled, deliberative, conscious cognition ('System 2') [197, 204].

The compositionality of human cognition provides a powerful ***learning bias*** (or ***inductive bias***) that pushes learning towards compositional analysis of novel task domains. Such a bias is needed for strong compositional generalization to emerge after learning from only a modest amount of experience (see discussion of Fig. 4, p. 37 and Fig. 7, p. 46 below). Thus compositionality holds the key to addressing two of the three fundamental problem areas facing current AI: limited compositional generalization (1*a*), p. 5, and extreme data inefficiency in learning (1*b*).

Compositional encodings have traditionally been formalized as intricate arrangements of symbols—***symbol structures*** like those we use to represent expressions in algebra or formal logic [153]. To illustrate with an extremely simple example from language, the symbols un, lock, able can be composed to form unlockable in two ways. In one arrangement, the symbols form the structure [un [lock able]]: to get the meaning of this structure, we first compose the meanings of lock and able to get the meaning of lockable, which we then compose with the meaning of un, yielding "not able to be locked". The structure [[un lock] able] is different: it means "able to be unlocked" because the same symbols are composed differently.

Compositionality has proved to play an important role in virtually all areas of cognition, extending far beyond language [75]. In planning, for example, plans to achieve a goal comprise sub-plans to achieve sub-goals; in reasoning, the derivation of a conclusion is the result of smaller derivations that derive subordinate conclusions [68]. In these and many other problem domains, traditional AI (and traditional cognitive theory) have made extensive use of compositional structure formalized as symbol structures [153, 172, 177].

In this context, symbols are abstract entities that are by definition inherently *discrete*: unlike real numerical values, they cannot be modified by arbitrarily small amounts (although physical drawings of what we perceive as symbols can be). The identity of a symbol, and its position in a symbol structure, are all-or-none: a symbol either is an A or it isn't; it either occupies the first position of a symbol sequence or it doesn't. Symbol structures admit no shades of gray. We will see, however, that compositional structures need not be formalized as symbol structures—they need not be discrete. It is the discovery of this surprising fact that gave rise to the new work presented here.

Thus formal theories of the human brain have given us neural computing, while formal theories of the human mind have given us compositional-structure processing, historically in the form of symbolic computing. Clearly, symbolic and neural computers are profoundly different (see Fig. 1, p. 9). Yet, somehow, the computer in our heads apparently is simultaneously a neural computer and a compositional-structure-processing computer.

How can this be? This is the **Central Paradox of Cognition** [211].

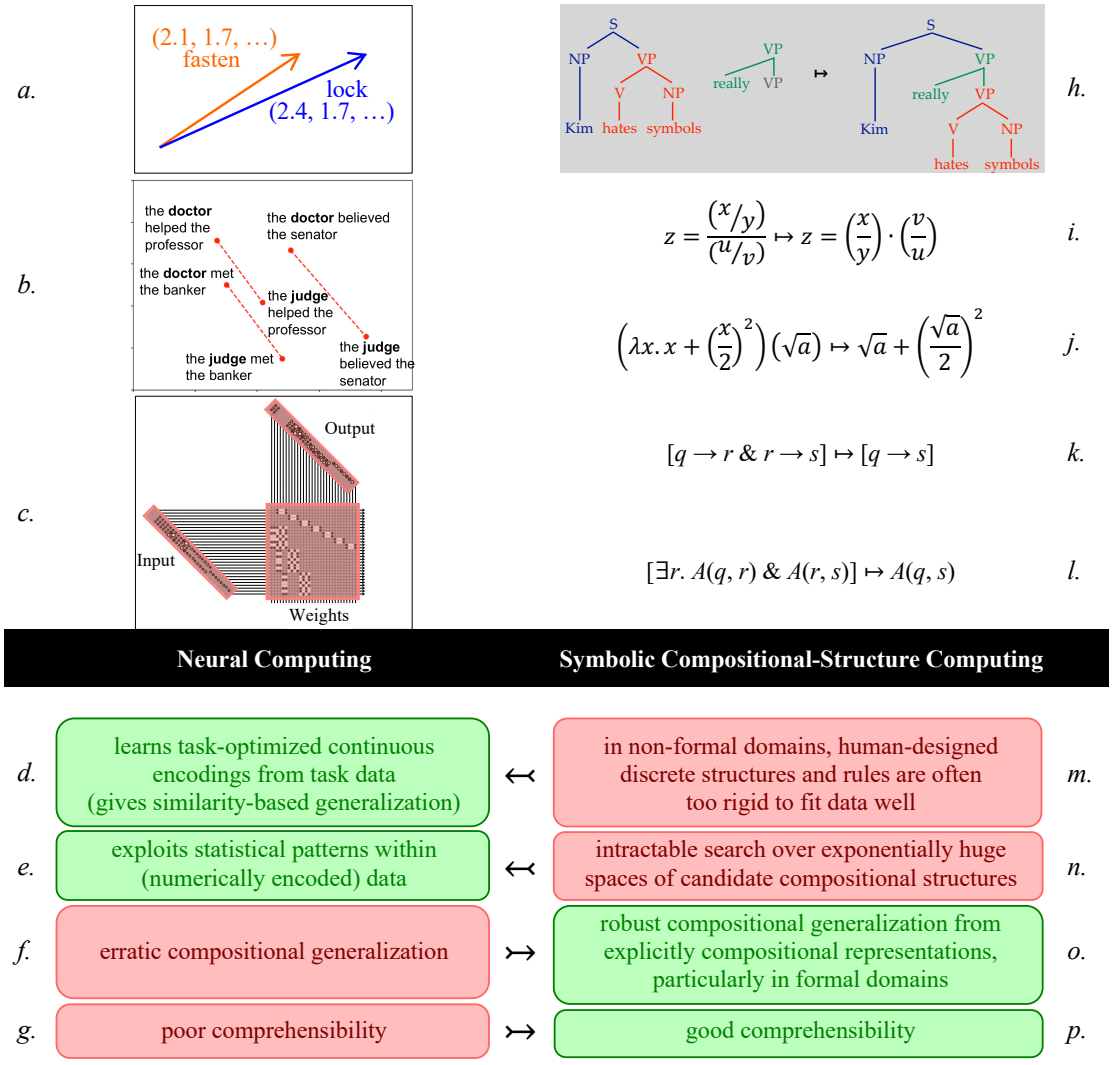

Figure 1: Neural and symbolic computing—Strikingly different (top), affording complementary strengths (bottom).

Figure 1: Neural and symbolic computing—Strikingly different, affording complementary strengths.

Left Top: Neural computing. Information is encoded as continuous numerical vectors, each a list of the activation levels of a layer of model neurons. Information is processed in neural networks by spreading activation from input vectors through weighted connections (model synapses) (***c***). Similar input vectors (***a***) yield similar output vectors: this is similarity-based generalization (***d***). For vector encodings of word sequences, relative position in vector space reflects structural similarity (***b***).

Right Top: Symbolic computing. Information is represented as arrangements of discrete symbols into structures (***h***), processed by functions mapping input symbol

structures (left of $\mapsto$) to output symbol structures (right of $\mapsto$). These functions are defined over variables: for algebraic structures, the ratio-of-ratios inference rule in ***i*** takes any input structure that matches the left side (with variables x, y, u, v replaced by specific symbol structures, potentially complex) and constructs the corresponding structure matching the right side; similarly for the function-evaluation instance in ***j***, where the function $f(x) = x+(x/2)^2$ is evaluated with $x = \sqrt{a}$; the implication-chaining inference rule ***k*** for manipulating logical structures: *q implies r and r implies s entails q implies s*; and the inference rule ***l*** for a transitive relation A such as *above*: *there exists r such that q is above r and r is above s entails q is above s*. The functions in ***h*** (tree adjoining [92]) and ***j*** (β-reduction in the λ-calculus [33]) are crucial to symbolic linguistic theory's formalization of compositionality for human syntax [91] and semantics [148], respectively.

Bottom. ***d-g, m-p*** identify strengths (in green) and limitations (in red) of each type of computing (see §1, p. 14 for explanation). A limitation of either type of computing may potentially be overcome by a strength of the other (arrows $\leftarrow$,$\rightarrow$). For instance, symbolic computing's often prohibitively long searches through huge spaces of possible symbol structures to find problem solutions (***n***) can be replaced by neural computing's rapid calculation of predictions via statistical inference (***e***). But merging the strengths of both types of computing requires solving the Central Paradox of Cognition: how can the computer in our heads be formally describable by both the left and right columns, given their fundamental differences?

A resolution of this puzzle is provided by a current theory positing that cognition is realized in neurocompositional computing [210]: computing with encodings each of which simultaneously respects both the Continuity and Compositionality Principles. This theory crucially relies on viewing the human cognitive computer at multiple scales. At a small scale, it is a neural computer. At a large scale, exactly the same machine is a compositional structure processor [77]. Recent research—comprising NECST and related theories—provides a mathematical account of exactly how this can come about [203]. (Visualizations will be provided in Fig. 2, p. 20 and Fig. 3, p. 25.)

This neurocompositional theory places cognitive science among the many other sciences that study system organization at multiple scales—most dramatically, biology. In physics too, the same material can be characterized at a large scale—by bulk-matter properties like temperature—and at a small scale—by molecular properties like kinetic energy. In computer science, the same machine is simultaneously, at a large scale, our familiar virtual machine offering us folders of documents and images and, at a small scale, a huge collection of 0s and 1s. In cognitive science, a cognitive state can be characterized by its large-scale compositional structure—perhaps containing something like [[un lock] able] as a part—and the same state can also be characterized at a small scale as a particular numerical pattern of neural activity. We will see in Sec. 2, p. 24 that, as in physics and computer science, in cognitive science, the cross-level linkage is fundamentally mathematical.

0.2 The anatomy of machine intelligence

How do AI systems relate to human cognitive computing? Many contemporary AI systems emulate the small-scale organization of cognition: they are entirely neural computers. Traditional AI systems, on the other hand, emulate the large-scale organization of cognition: they are compositional structure processors, formalized discretely, with symbol structures. The work presented below derives from the hypothesis—defended in Sec. 1, p. 14—that, to approach human-level intelligence, ***an AI system needs to fully emulate the organization of the human cognitive computer, both at small and large scales***.

This requires a type of higher-level organization that is new to AI. To coexist with their small-scale neural organization, which is continuous, the large-scale compositional structures must be formalized in a continuous framework. Importantly, these computational structures can be usefully *approximated* as discrete symbol structures—but a precise description requires that they be formalized as a new type of encoding: ***continuous structures***, built not of symbols, but of large-scale neural activation patterns. We will provide evidence that the flexibility inherent in a continuous formalization of compositional-structure processing addresses a fundamental limitation of traditional symbolic theories of machine and human intelligence: the rigidity of discretely formalized compositionality.

Below we will argue that the most successful models that have driven the recent deep-learning revolution derive their power from limited forms of compositional high-level structure. This is **first-generation (1G) neurocompositional computing**. A prime example is the ***Transformer*** architecture [229, 237, 120].

As we explain below (§2.2.2.2, p. 36), Transformer networks are distinguished by two types of large-scale structure: one is literally wired in (sequential structure), the other arises from special small-scale capabilities that enable the network to build larger-scale compositional structures of one specific type (graphs) [67].

Although there are important open questions here that we take up below, the success of the Transformer suggests that, to develop still more powerful network types which deploy stronger compositional structures—of the many types observed in human cognition, robustly across cognitive domains, without requiring astronomical quantities of data for learning—a new generation of AI models is needed. Carrying the limited compositional-structure-processing enhancements of the Transformer considerably further, these new models possess general low-level network capabilities that enable them to construct an open-ended class of large-scale compositional structure types, and the capabilities necessary to exploit this structure explicitly via the Compositionality Principle: the capability to decompose complex structures into their parts, and to produce complex structures by composing together simpler structures. Any particular model of this new generation can *invent its own* ***types*** *of large-scale structures through deep learning*; as we will see, this means inventing compositional structures that are *optimized* for the model's performance on its particular tasks. These new models we present rely on a more full-blown realization of neurocompositionality: they comprise **second-generation (2G) neurocompositional computing**.

As mentioned, a specific technical framework for achieving neurocompositional computing is presented here: Neurally-Encoded Compositionally-Structured Tensor (NECST) computing. We present evidence from theory and applications for the far-reaching po-

tential of a NECST generation of AI systems. As discussed below in §2, p. 24, we focus on NECST computing because, while other fruitful frameworks for neurocompositional computing have been developed, many of these turn out to be compressed forms of NECST encodings. Development of NECST is thus a good branching off point for studying other frameworks for neurocompositional computing.

We also discuss below (§1.2.2, p. 19) how neurocompositional computing relates to existing *hybrid neuro-symbolic computing*, in which only parts of the system have the small-scale organization of neural networks; all compositional structure is relegated to the remaining parts, which are standard discrete symbolic computers. This is *Neurons-in-Structure* organization, in contrast to the *Structure-in-Neurons* organization of neurocompositional machines (illustrated in Fig. 2, p. 20).

Bringing AI systems into closer alignment with human cognition provides opportunities for important additional benefits: these systems have the potential to be more human-comprehensible than current AI models, which are notoriously opaque [23]. Neurocompositional computing thus addresses the third problem area of contemporary AI: impenetrability to human understanding (1c), p. 5. Improved understanding of the internal encodings of neural networks could enable us to bias networks performing a particular task towards learning particular types of compositional structure that have been shown to be useful by our theories of the task. Sufficiently thorough understanding of a network should allow us to diagnose internal errors and to even intervene and directly alter the network's internal encodings to partially control its outputs. In Sec. 3, p. 40, we present early evidence that such new types of human/neural-computer interaction can indeed be realized through neurocompositional models.

0.3 Main claims: Continuous compositional structure within neural states

In summary, the main claims argued for in the remainder of this tutorial are these.

(3) **Section summary: Main claims of the tutorial**

a. Despite dramatic recent progress, contemporary AI systems still fall significantly short of general human intelligence in a number of respects, including compositional generalization and learning-data efficiency; in addition, these systems suffer from extreme impenetrability to human understanding (1), p. 5.

b. Traditional AI, although stronger in these respects, also suffers from its own major limitations: see point e below.

c. To overcome all these shortcomings, AI needs to emulate the organization of human cognition, at both small and large scales. This means AI systems that are neurocompositional computers: at a small scale, they are entirely neural network computers; at a large scale, they are computers which process encodings that possess structure that is simultaneously compositional and continuous.

d. Neurocompositional computing addresses the three weaknesses of AI identified in (1). It derives improved compositional generalization from its large-scale organization as a compositional structure processor. This organization provides a learning bias that pushes towards treating tasks compositionally; such a bias is needed for improved data efficiency, enabling modest-sized training sets to

produce strong compositional generalization. And emulating the large-scale organization of human cognition reduces the impenetrability of AI models.

e. Because the large-scale compositional structure of encodings is realized in continuous numerical vectors, neurocompositional computing also addresses two key limitations of traditional AI discussed below (§1.2, p. 18): inflexibility of discrete symbolic encodings and intractability of inference over exponentially large spaces of symbol structures.

f. To achieve neurocompositional computing, at the small scale, new capabilities need to be built into neural networks: capabilities that enable, at a large scale, basic composition and decomposition operations with which deep learning can invent and process a wide variety of compositional structures.

g. First-generation neurocompositional computing—e.g., the Transformer—has already greatly advanced AI. Second-generation neurocompositional computing, developed in the work presented here, takes these advances considerably further. The mathematical techniques we deploy for this—NECST computing—are derived from contemporary cognitive science.

h. NECST is supported by strong theoretical results and by successful results from a diverse collection of early models addressing a range of applications.

i. The incorporation of more human-like organization makes NECST models more interpretable, instructible, and controllable than the inscrutable deep learning models that now dominate AI.

The structure of the tutorial is given in (4)

(4) **Tutorial outline**

a. In this section, we have defined *what* neurocompositional computing is (§0, p. 4). In the following sections we argue:

b. *why* neurocompositional AI is needed (§1, p. 14);

c. *how* neurocompositional AI can be realized using NECST computing (§2, p. 24); and

d. that early NECST AI systems show how neurocompositional AI improves compositional generalization and benefits from its enhanced *comprehensibility* (§3, p. 40).

e. We close with an overall summary (§4, p. 50) and then

f. look ahead at the future of NECST-generation AI (§5, p. 53).

g. Appendix A provides more detail on the foundations of NECST encoding of compositional structure (§6, p. 55).

h. Appendix B provides details of the reported experiments on compositional generalization in a simple compositional task (§7, p. 57).

i. Appendix C illustrates how continuous NECST processing constructs the compositional grammatical structure of a sentence heard a word at a time (§8, p. 60).

The weaknesses in neural computing that derive from lack of compositionality and related aspects of symbolic computing have been emphasized in the influential work of Jerry Fodor and Zenon Pylyshyn [22, 53], Gary Marcus [128, 131] and others, and they have been a focal point of contemporary deep learning research by Marco Baroni [6, 102], Yoshua Bengio [11, 146, 185], and many others [56, 60, 69, 78, 82, 83, 104, 118]. The NECST research program we present has greatly benefited from the foundational work of these researchers, although there is of course no implication of their endorsement of the claims made here.

0.4 Two analogies

In this tutorial, we offer two analogies that may help convey the subtle sense in which continuous compositional structures are "in" the neural activation patterns of NECST systems. In the deepest analogy, words reside in a NECST encoding of a sentence as notes reside in music. We defer presentation of this analogy until explanation of NECST encodings in §2, p. 24 (Fig. 3). A less faithful, but simpler, analogy is provided by the spiral structure of a galaxy. Limited as we are to viewing the stars in the Milky Way from "nearby", we can't see directly the large-scale structure of our galaxy, although an extraterrestrial viewing it through a telescope from light-years away could clearly identify the barred spiral structure. To see the large-scale compositional structure of NECST encodings, we too need to look at them from afar, through a metaphorical telescope (see Fig.2, p. 20)—in reality, a certain type of well-specified mathematical analysis of large-scale structure that actually has much in common with the computations that must be performed to identify notes in the acoustic waveforms of music. The parts that compose together to form a NECST large-scale compositional structure—perhaps including encodings of un, lock, able—are in fact encoded as different spatial "waves" of neural activation that superimpose, exactly like simultaneous sounds, to encode the structure as a whole.

1 Why neurocompositional AI?

This section (§1) presents our arguments for *why* a new generation of AI based on neurocompositional computing is needed; the following section (§2, p. 24) then turns to *how* this new type of computer can be realized using the NECST framework. Here in section §1, we take up, in turn, the two principles that define neurocompositional computing: individual encodings must respect both the Compositionality Principle and the Continuity Principle. We discuss what goes wrong when each principle is flouted, and why those weaknesses turn to strengths when the principle is respected. Also discussed is the need for providing networks with new capabilities for assembling and disassembling compositional structures. Along the way, we identify important open questions bearing on the future developments needed to narrow the gap separating machine and human intelligence. The impatient reader may choose to skip to the summary of section 1 (5), p. 23 and continue from there.

1.1 Why respect the Compositionality Principle?

The most fundamental requirement for neurocompositional computing is compositional encodings. What problems arise from the absence of compositional structure in encodings? First and foremost, inadequate compositional generalization: the ability to correctly process any novel input that is composed of familiar parts [6, 96, 103].

In standard neural networks, encodings have no built-in compositional structure. To illustrate the limitations this can bring, consider the seemingly trivial task of reproducing a sequence of digits: having been given the input 98052 one digit at a time, we want a standard neural network model—call it CopyNet-0—to produce the output 98052 one digit at a time. Using deep learning methods discussed below, CopyNet-0 extracts its knowledge from a training set of example pairs of its desired input-output behavior, such as 98052 → 98052. Suppose that, with malice aforethought, we impose a constraint on the training set such that CopyNet-0 never sees training examples containing the digit 1 in position *1*, nor 2 in position *2*, and so forth. Simulations discussed below (Fig. 4, p. 37) show that CopyNet-0 can learn to perform perfectly on all its training examples, and on novel test examples that follow the constraint imposed on its training set, yet still perform very poorly on sequences containing any digit in an unfamiliar position—even though it has seen each digit in all other positions, and seen all other digits in any given position. This is a blatant failure of compositional generalization in what may be the simplest possible task.

A non-trivial task on which compositional generalization was tested is the transformation from a sentence to an expression in logic that denotes its meaning, for a set of English sentences produced by a formal grammar. The two types of neural network tested (including the Transformer) mastered the complex examples in the training set, but failed a range of tests of compositional generalization. For example, although the models could correctly process a training input like Sandy loves Kim, they could not do so for a novel test item such as Sandy loves Robin even though they had learned the meaning of Robin in isolation [96]. In contrast, 2-year-old children already exhibit similar types of generalization [225].

Failures of compositional generalization can reveal the shallowness of understanding even in models that are deemed to have mastered their task. In the natural language inference task, two sentences are presented and a model must indicate whether the second follows logically from the first [234]. On a standard measure of success, the landmark Transformer language model BERT [46] achieved 84% correct. But when tested on new, carefully-designed questions, the model's performance plummeted to 4% [136]. For instance, BERT asserted that the truth of the sentence the judge chastised the lawyer entails the truth of the lawyer chastised the judge. The new questions were not difficult for humans to answer, but they were constructed to foil the model if it were responding based on specific superficial properties like whether the words in the two sentences were the same, ignoring ordering—which is critical in the new test cases. The standard competitions among AI research teams for the highest score on fixed test sets is an evolutionary process that optimizes success on the exact test; performance on alternative, equally valid tests of the desired knowledge can be vastly lower. With this method of evolving champion models, the research field is in effect "teaching to the test", which notoriously can render test results meaningless, giving the false impression of a level of

mastery indicative of significant compositional generalization. On the original test, the model was often getting the right answer—but for the wrong reasons.

1.1.1 Robustness of compositional generalization

To what extent can current neural networks trained with deep learning in fact achieve compositional generalization? This is a fundamental issue that is crucial to the future of AI [55]. It is currently an open question, but on balance, such generalization is a weak area of current AI systems [7]. The degree of compositional generalization that such systems do display is dependent on enormous quantities of training data that adequately exemplify the exponentially many ways parts can compose in the given task. In neurocompositional computing, one function of compositionally-structured encodings—and the operations needed to exploit this structure—is to give these models a learning bias that pushes towards treating a task compositionally. Such a bias is needed to enable strong compositional generalization from training sets that are not gigantic.

This question of the degree to which current AI models display compositional generalization is receiving increasing attention, and a complex picture is only starting to emerge. A network's compositional generalizations are of course highly sensitive to the range of examples it learned from, and carefully augmenting the training set can improve generalization [142]. But overall, compositional generalization in networks has proved highly erratic (Fig. 1*f*, p. 9)—for example, it turns out to be highly sensitive to the random values of connection strengths at the start of learning [135]. There are striking failures but also impressive successes. An example of an evolving model that displays both these features is the gargantuan neural language model GPT, which has generated numerous well-publicized long stretches of text in highly fluent English [20, 174]—composing new combinations of parts it has extracted from the astronomically-sized dataset it was trained on, sometimes in ways that suggest the model has deep understanding of what it's saying. But it also displays shocking failures to compose these parts in coherent ways, revealing how much apparent understanding is contributed by the human reader, not the model [48, 130].

Another case is a Transformer network we will call IntNet [106], which learns to compute indefinite integrals of functions given in symbolic form, generalizing from the complex examples it was trained on to novel complex functions built of parts seen in training: an impressive degree of compositional generalization. But even here, what IntNet has learned is not as general as we imagine it to be. It does extremely well on novel problems that are created using the same procedure that generated the examples it was trained on, but not on novel problems generated by a different procedure [45]: apparently, it has learned to process the particular compositional patterns it saw during training, but many other equally relevant patterns of composition cannot be processed correctly.

How does the situation change when encodings *do* possess appropriate compositional structure? Returning to the simple digit-sequence copying network CopyNet-0, its clear failure of compositional generalization surprises us because we assume—since the model performs this ridiculously simple task perfectly, on a huge range of inputs—that it has learned a general procedure that will reproduce *any* input.

Now consider a hypothetical (perhaps symbolic) model CopyComp whose internal encoding of any input, e.g., 1234, has the following two-part compositional structure: the first part is the encoding of the first digit of the input, 1, and the second part encodes the

rest, 234. (The second part is itself a smaller compositional structure embedded within the larger input sequence.) Then the model can perform its task by taking the encoding of the first part, outputting that same encoding, and then starting the process over, now taking the second part as its input. Unlike CopyNet-0, CopyComp is guaranteed to correctly reproduce any input.

Consistent compositional generalization is a strength of traditional AI because it operates over encodings with explicit compositional structure using explicitly compositional operations (Fig. 1*o*, p. 9). If a symbolic AI system's knowledge includes a grammar, it necessarily can process any of the typically infinite varieties of grammatical sentences; if its knowledge includes procedures for certain steps of logical inference, then it necessarily can infer, from a given set of premises, any conclusion that follows from those premises by some (potentially extensive) chain of such steps.

1.1.2 Partial compositional structure

What conclusions should be drawn from the partial compositional generalization observed in networks like CopyNet-0 which (unlike Transformer networks) have no built-in capabilities for building structured encodings? Does it mean that such generalization does not require compositional encodings? Or do these networks in fact have partially-compositional encodings at a large scale, where this structure arises purely through learning, without the benefit of built-in structure-building capabilities? These are further urgent questions on which the future direction of AI depends.

There is some evidence that, when properly viewed, the large-scale organization of the encodings learned by current language-processing neural networks does bear remarkable resemblances to the symbolic compositional structures posited by traditional linguistic theories [223]. This evidence is strongest for Transformer models which, as we explain below (§2.2.2.2, p, 36), are 1G neurocompositional systems that are provided with low-level capabilities for building compositional structure: in this case, structure of a particular type—a graph, in the computer science sense: a structure which places links that, in the encoding of a sentence, connect pairs of words that network computation has determined to be related. In the Transformer language model BERT, the learned graph structure has significant commonalities with linguistic-theory relations between words, but unlike traditional graphs, the Transformer's graphs are not discrete but rather continuous structures: between a given pair of words, a link is present to a continuously-variable degree—its presence is not all-or-none. The general power provided by such continuity will be taken up in the next subsection (§1.2, p. 18). In the particular case of BERT, the learned continuous encodings of words have a striking large-scale organization within their vector space: when properly computed, the distance between the vectors that encode two words in a sentence correlates with the distance between those words in the linguist's compositional phrase structure (tree diagram) of the sentence [127].

1.1.3 Compositional processing?

In what ways do the learned implicit large-scale partial structures embedded in models like BERT contribute to their generalization capacity? This is another important question that is currently open. Because of the still-intractable problem of understanding how particular internal features of a network causally affect its performance, it is unknown whether the compositional aspects of the large-scale organization extracted by analysts

from some networks can be, and in fact are, used by these networks [9, 85, 176]. (Section §3.1, p. 40 discusses compositional processing in MATHNET, a neurocompositional model for math problem-solving.)

In contrast, the techniques we propose below endow NECST networks with the general capability to create explicitly-compositional structures of a wide range of types, and to directly compose and decompose these structures during the performance of their tasks.

In the next section (§2, p. 24), we discuss evidence that, under certain conditions (which demand extensive training examples), networks without the built-in structure-building capability of Transformers also learn encodings with some degree of large-scale compositional structure. Nonetheless, as we discuss below, we consider it likely that, to achieve strongly compositional encodings and thereby achieve robust compositional generalization without massive training datasets, networks need built-in capabilities for inventing, and computing with, general structure.

We now pass from compositionality to the second requirement on neurocompositional encodings: continuity.

1.2 Why respect the Continuity Principle?

To obtain the benefits of compositionality, why not just do what has been done for thousands of years: formalize compositional structure discretely with symbols [162]? Why develop *neuro*compositional computing, which requires a new, continuous formalization of compositionality, embedded within neural computing?

1.2.1 Flexibility

What limitations arise from compositional structure that is discrete? First of all, inflexibility of encodings. Theories of human cognition that encode information in discrete compositional structures have provided unparalleled understanding of most cognitive domains [236], but they have often encountered difficulties, as have the traditional symbolic AI models using such structures [16, 21, 97, 132]. There are important cognitive domains in which information has exact discrete compositional structure, by construction: these are artifacts of deliberate human design such as mathematics, logic, programming, and many games. But outside such *formal* domains, for instance in natural language, discrete compositional descriptions have proven to be seriously limiting. Discrete structures seem too rigid for non-formal domains[4]—or perhaps the needed degree of flexibility can only be approximated with discrete structures of greater complexity than theorists have been able to craft. In any case, these limitations are a primary factor in the shortcomings of traditional symbolic AI. In natural language, the subtle variations in meaning of even a single word surpass the capabilities of symbolic structures: having to create a separate symbol for each of the distinct meanings listed in a dictionary merely begins to capture the magnitude of this problem [171].

In contrast, the continuous vector space of neural encodings has proved much more adequate (Fig. 1*a*, *d*, p. 9) [139, 163]. Machine processing of natural language has been revolutionized by vector encodings of words [221], particularly those that are sensitive to their context [164], like those produced by BERT [156]. Well beyond language, vector

[4]In this tutorial we refer to a problem domain as a 'formal domain' if its states and state transitions are *defined* by discrete formal rules, such as inference with algebraic expressions or execution of computer programs.

encodings have fueled major advances in virtually all areas of AI [35]—even in formal domains, as we see below (§1.2.3, p. 21).

Continuous vector encodings derive their power in part from their support of generalization based on vector similarity: what is learned about one vector encoding generalizes to similar encodings, located nearby in vector space [71, 73]. For example, because of their similar distributions in training text, the words fasten and lock will tend to have similar learned encodings (Fig. 1*a*, p. 9), enabling a language-processing model to leverage its extensive experience with the word lock when processing the rarer word fasten. The structure of the vector space can capture other types of relationships as well, such as having a consistent offset that represents a systematic difference between pairs of sentences (Fig. 1*b*). But most important to the power of vector encodings, discussed in detail in §1.2.4, p. 22, is what their continuity facilitates: optimization for task performance.

1.2.2 *Neurons-in-Structure* vs. *Structure-in-Neurons* computing

It is the limitations arising from the rigidity of discrete symbolic formalization of compositional structure that lead us to argue that *general* human-level intelligence will require neurocompositional computing rather than a style of *neuro-symbolic computing* currently being pursued by some AI researchers. In this approach, a symbolic computer program is designed to build and process discrete compositional structure, and then, for some of the functions executed within this program, the internal symbolic subprograms that perform these functions are removed and replaced with neural sub-computers that perform similar functions (see Fig. 2, left half). We call this this type of computing *Neurons-in-Structure*; these are *hybrid systems* in which there are distinct symbolic and neural parts of the machine. The small-scale organization of such computers is neural for only certain components of the machine, in contrast to neurocompositional computing, where the small-scale organization of the entire machine is neural and so compositional structure must be embedded within neural activation patterns: this is *Structure-in-Neurons* computing (Fig. 2, right half; see also Fig. 3, p. 25).

In Neurons-in-Structure computing, all compositional processing is carried out with discrete symbolic computing, so as in traditional symbolic AI, for informal problem domains, the inflexibility of symbolic compositional structures is a serious limitation. Symbolic formalizations of compositionality are well-suited, however, to formal problem domains (including synthesizing computer programs, proving theorems, and playing many games) and as we discuss next, Neurons-in-Structure systems have produced impressive results in such domains. Because non-formal domains are central to general intelligence, however, we argue here that human-level *general* intelligence calls for Structure-in-Neurons style—neurocompositional—computing.

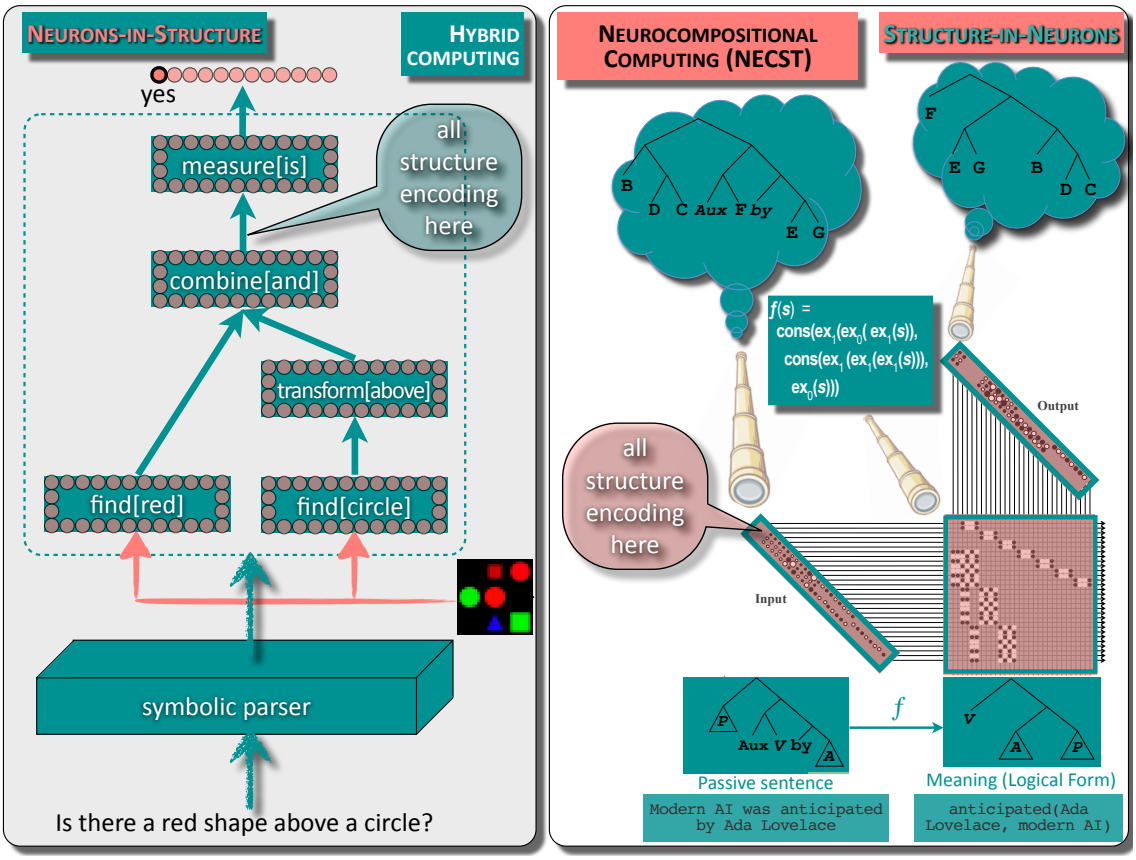

Figure 2: Neurons-in-Structure vs. Structure-in-Neurons computing.

Figure 2: Neurons-in-Structure vs. Structure-in-Neurons computing.

Left: Neurons-in-Structure—hybrid—computing. In this approach, a symbolic computer program is designed to build and process discrete compositional structure, and then, for some of the functions executed within this program, the internal symbolic subprograms that perform these functions are removed and replaced with neural sub-computers that are trained to perform similar functions. In a visual question-answering model [3], a symbolic parsing algorithm takes as input a question about an image and constructs a program, here drawn as a graph: teal boxes are operations, with data flowing along arrows. The data are activation vectors and each operation (box) is performed by a separate neural network: the find[red] operation is a network that depresses activation representing non-red regions of the image. All compositional structure is encoded in the symbolic algorithm graph; the embedded neural networks do not themselves encode or process the large-scale structure of the operations being executed.

Right: Structure-in-Neurons—NECST—computing. In this approach, a neural computational system manipulates activation vectors that have special large-scale structure that mirrors the compositional structure of the information they encode. In this model, LFNET, each neuron's activity level is depicted as the size of a disk:

white/positive or black/negative [114]. The input vector is a sum of vector embeddings, each of which represents a constituent in a structure. The input shown is the encoding, in a vector space, of a passive English sentence (e.g., Modern Artificial Intelligence was anticipated by Ada Lovelace), with a tree structure given by the diagram of the sentence's phrases nested inside one another. Analogously to identifying the temperature, pressure, and entropy of a gas of molecules, or the spiral form of a galaxy, recognizing that the input vector encodes this structure requires the analyst to observe it at a larger scale than that of individual neurons (shown schematically by the telescopes and teal thought bubbles): the global structure of the vector must be computed, identifying which symbol-encoding vectors have been superimposed to form it (see also Fig. 3, p. 25, right half). Similarly for the output vector, which encodes the LF or logical form (meaning) of the input sentence (e.g., anticipated(Ada Lovelace, modern artificial intelligence)). The connection weights governing the flow from input to output also form a pattern with special structure: an appropriate larger-scale analysis reveals it to be the structure-manipulating function f (central teal box). This is Neurally-Embedded Compositionally-Structured Tensor (NECST) computing. All structure is internal to the activation and weight patterns themselves.

1.2.3 Tractable inference

Does compositionality formalized using discrete symbol structures create problems other than inflexible encodings? Yes: inferring outputs from inputs is often computationally intractable; this is another serious limitation of traditional AI, one which applies even in formal domains. While the ability to insert symbol structures inside other symbol structures *ad infinitum* may enable strong compositional generalization, it also creates exponentially huge spaces of candidate problem solutions over which symbolic algorithms must search to find an actual solution (Fig. 1*n*, p. 9). For example, symbolically encoding board configurations in the game of Go yields more structures than there are atoms in the universe [155]. The prohibitively long search times of traditional AI systems for performing symbolic integration enables neural INTNET (§1.1.1, p. 16) to outperform them, even in such a highly formal domain as symbolic calculus [106].

In contrast, as we see next in §1.2.4, deep learning's exploitation of statistical relations among optimized vectors that compactly encode examples from a problem domain has proved remarkably capable of quickly producing problem solutions that are excellent (Fig. 1*e*), although they may fall somewhat short of the perfect solutions a symbolic algorithm might have found had it started searching at the Big Bang. Neurons-in-Structure models exploit this strength of neural computing, enabling them to make major progress on AI in formal domains. Well-publicized breakthroughs have come from such hybrid models, in which search in huge symbolic problem spaces is guided not by symbolic computing but by predictions of promising search directions by neural networks trained with deep learning by observing a large number of successful and unsuccessful searches [192]. This has produced super-human models for playing games, like Go, that were thought, until very recently, to be well beyond the reach of AI in the foreseeable future [193]. Neurons-in-Structure models are also making major strides in automatic generation

of computer programs that solve a problem stated in English [121], or programs that compute a function partially specified with a small set of example input/output pairs [47]. While we have argued that neurocompositional, Structure-in-Neurons computing is needed in the long term for modeling general human-like intelligence, we also acknowledge the value of Neurons-in-Structure computing for its ability to make progress in the immediate term on AI in formal problem domains: it can exploit today the full power of symbolic computing in such domains [17, 230].

1.2.4 Optimized compositional encodings

Neurocompositional computing meets the requirement of continuous encodings by using vector encodings consisting of a list of continuous real numbers. These numbers constitute the encodings of the input, the output, and the intermediate information used to compute the output. Where do these numbers come from? Typically, from optimizing the encodings for performance of a task—through deep learning.

The problem solved by deep learning within a model performing a particular task is this: how can inputs and outputs from the task be encoded in numerical vectors such that the numbers in the output vectors can be predicted from the numbers in the input vectors via the numbers in appropriately designed intermediate vectors? Once a computational task is encoded in numerical vectors, solving it becomes a problem in statistical inference. A neural network is a complex statistical model; learning is estimating its many parameters (primarily, connection weights)—fitting the model to the training data. During learning, the network can compute and store in its weights the relations between the numbers in the vector encodings produced when processing the training examples; after learning, during testing, it can use these learned relations to infer an output vector given an input vector that encodes a novel input from a test set.

Thus there are two problems to solve. How should task information be encoded in numbers? And what are the interrelations among those numbers? The extraordinary power of deep learning is that it allows these two problems to be solved simultaneously: it figures out how to encode information in numbers such that the relations among those numbers optimize the predictability of the output numbers given the input numbers. To do this, it relies on a function that quantifies, at any point during learning, the total error made by the network in processing the set of training examples. For each parameter to be learned—principally each connection strength—it computes the rate at which this error would decline if that parameter were increased by an arbitrarily small amount: the negative of the derivative of the error with respect to the parameter. It then changes each parameter by a very small amount proportional to this negative derivative: this lowers the error, changing most those parameters that have greatest influence on the error. This optimization process drives the network's parameters to produce lower error while at the same time discovering the encodings of information that achieve this [182]. Note that the very existence of the error derivatives that drive deep learning depends crucially on the continuity of neural computing.

The astounding discovery that launched the recent leap in AI is that, by minimizing *a single number* that measures the overall error made by a network with trillions of parameters when processing trillions of bytes of data—and precisely *when* the size of the network and the training set are both enormous—this simple learning procedure, with no human input on how to perform a task, can produce AI systems that dramatically outperform the vastly more intricately designed, richly human-informed AI systems of

previous generations (Fig. 1*d–e*, p. 9).

What links the properties of continuity, achieved with real numerical vectors, and optimization, then, is statistical inference. The encodings within the network, designed by deep learning, reflect a sophisticated analysis of the statistical patterns within the input/output pairs of a given task. Neural networks compute outputs from inputs by statistical inference, rather than the discrete logical inference deployed by many traditional AI methods.

Statistical inference has also been used in non-neural AI systems that compute the probabilities, given an input, of alternative discrete symbol structures that encode intermediate information used to compute the output; these symbol structures are generated with symbolic computing, and do not reside in neural encodings [89, 108]. Such probabilistic symbolic methods have in common with the Neurons-in-Structure approach to neuro-symbolic computing that the structure is discrete and housed within subsystems capable of symbolic computing. In contrast, Structure-in-Neurons systems use no symbolic computing, with structure residing entirely inside neural activation vectors—continuous, rather than discrete, compositional structure. How this can be done is the topic of the next section (§2).

To summarize:

(5) **Summary of Section 1**

a. Current AI systems display erratic compositional generalization, whereas human cognition is characterized by widespread, robust compositional generalization.

b. To imbue AI with such capabilities, AI systems should emulate the large-scale organization of cognition: they should be compositional-structure processors.

c. Properly endowing networks with explicit capabilities for processing large-scale compositional structure with neural computing would provide them a powerful learning bias directing learning towards compositional analyses of the task, so that stronger compositional generalization can result from learning over less data.

d. Discrete formalization of compositional-structure processing has provided much insight into cognition, but, especially in non-formal problem domains, it brings fundamental limitations to traditional symbolic AI due to the inflexibility of discrete encodings and the intractability of inference over such encodings.

e. Continuous vector encodings in current AI models have provided powerful leverage for addressing both the problems of inflexibility and intractability. But while standard neural networks employ small-scale continuous encodings, they only possess limited aspects of the large-scale compositional structure that we have argued is crucial for strong compositional generalization.

f. To overcome the limitations of discreteness, *compositional structure itself* must be continuously encoded neurally. This calls for the Structure-in-Neurons approach of neurocompositional computing.

g. In the alternative Neurons-in-Structure computing, compositional structure is handled with symbolic computing and so is necessarily discrete; neural computing is relegated to only part of the hybrid system, and does not operate

compositionally. While this does not overcome the limitations of discrete compositionality, it does offer the full power of symbolic computing immediately, and has enabled much progress on modeling formal domains such as automatic computer program synthesis.

h. This section has provided arguments in favor of developing neurocompositional systems for AI, defined to be models in which information encodings have compositional structure and are continuous rather than discrete: models which possess built-in capabilities for creating and using this compositional structure.

i. Compositionally-structured encodings promise stronger compositional generalization than that of standard neural AI models, even given limited training data; continuity delivers more flexible encodings and tractable inference than does discrete symbolic AI; optimization during learning provides data-driven design of these continuous compositional encodings to maximize the effectiveness of statistical inference for task performance.

So how, in theory, can neurocompositional computing be realized? And does it work in practice?

2 How to realize neurocompositional AI? NECST computing

Neurocompositional computing arises from the discovery that compositional encodings can be formalized not just in the traditional way, with discrete symbol structures, but also in a new way, with continuous neural activation vectors. Several fruitful frameworks for neurocompositional computing have been developed, a number of them under the rubric of Vector Symbolic Architectures [57, 116]. Some use binary-valued neurons [95], some use spiking [50, 191] or oscillating [231] neurons, and many depend on certain random distributions of neuron values in encodings [167]. A number of these seemingly quite different methods turn out to be linearly compressed forms of NECST encodings [214], so NECST is thus a good starting point for studying other methods of neurocompositional computing. And NECST is among the simplest and most thoroughly-developed of the neurocompositional approaches. For all these reasons, we focus here on NECST as a framework for developing neurocompositional computing.[5]

This section gives a non-technical description of how NECST computing provides a continuous formalization of compositional-structure computing. We start with the special case of how discrete compositional structure is formalized in NECST, and then turn to the general case: continuous structure. For readers seeking more development of NECST methods, Fig. 9, p. 55, steps through the construction, introducing several details that are omitted from the simplified discussion contained in this section.

[5] Tensor Product Representations (TPRs), the encodings forming the basis of NECST computing, are introduced next (§2.1.1). A range of compression methods are used to derive other neurocompositional encoding systems from TPRs, some of them anticipated in psychological memory models [151]. While such compression produces smaller encodings (fewer neurons), it also introduces noise that interferes with exact compositional computing. The need to compress TPRs has often been exaggerated in the literature because the sizes of TPRs have been miscomputed, yielding values that are overestimated by orders of magnitude: [128, p. 106], 3–4 orders; [219, p. 607] repeated [39, p. 815], 6.5 orders. Controlling the noise in compressed TPRs requires increasing their size, so that (noisy) compressed encodings used in published models are typically actually larger than the full TPRs needed to encode the same information noise-free [208, p. 1112].

2.1 Neurally encoding discrete structure

In this subsection, we first present the NECST method for realizing discrete neurocompositional encodings; we then discuss evidence for the adequacy of this method. As previously mentioned, words are "in" NECST encodings of sentences as notes are "in" music: this analogy is presented in Fig. 3 through a visualization of the method developed in this section.

2.1.1 Continuous encoding of discrete structure

What are the most fundamental ideas of NECST? The first idea is to make explicit what is typically left implicit in notations for symbolic compositional structure: the 'roles' that define a type of structure by characterizing the positions within the structure where content—e.g., symbols—can be placed. This explicitly *disentangles* bits of content—the 'what'—from bits of form—the 'where' (see also [66, 70, 185]).[6]The second basic idea is to encode not just symbols, but also structural roles, continuously: as neural activation vectors.

Any type of discrete symbol structure can be formalized as a set of structural *roles*, which can be visualized as all possible positions within the structure that a symbol (or a smaller symbol structure) could occupy [153]. Although these roles are often left implicit in the notation used for symbol structures, they are crucial, and we make them explicit in our notation. The discreteness of symbol structures means that roles themselves are discrete: a symbol either fills a particular role, or it does not.

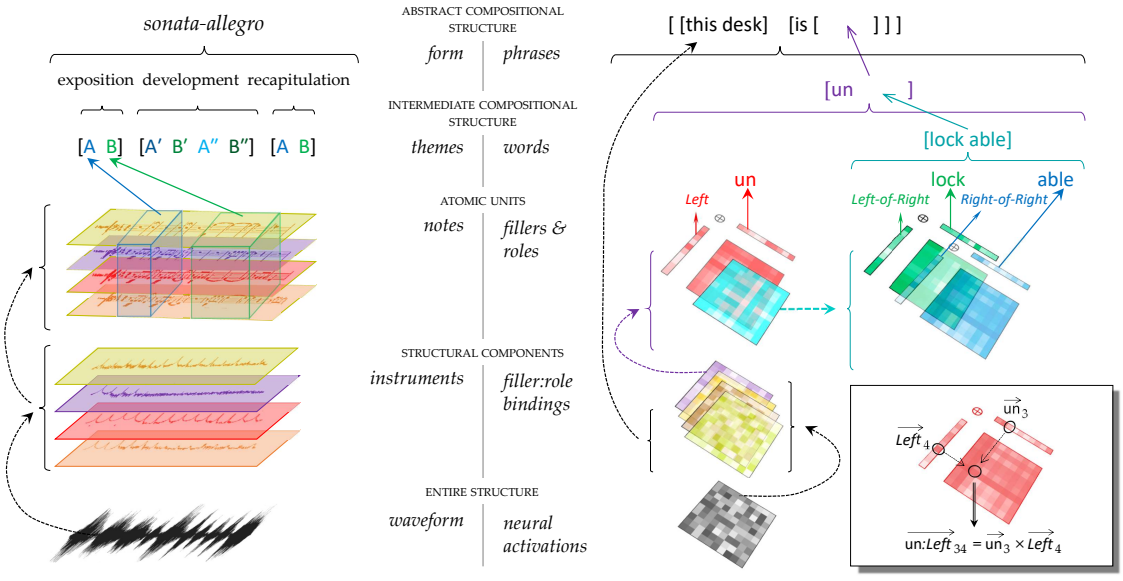

Figure 3: The levels of structure "in" the NECST encoding of a sentence (right half) are analogous to the levels of structure "in" a symphonic movement (left half).

[6]For evidence that the brain uses two distinct streams for processing structure and content, see, e.g., [175] p. 719, 721, and [233], which deploys tensor product binding (p. e6; Fig S4).

Figure 3: An analogy—Words are in neurocompositional encodings as notes are in music

How is the structure of a sentence encoded in a NECST activation pattern? The NECST encoding (right side) has a compositional structure analogous to that of a symphonic movement (left side). Words and phrase structure reside in a NECST encoding of a sentence in much the same way that notes and melodic themes reside in classical music.

Left: The levels of a symphonic movement, from sound wave to *sonata-allegro* form. Consider an orchestra conductor listening to the first movement of a classical symphony. At a small scale, the music is a single continuous sound wave striking her eardrums. But her highly skilled perceptual system enables her to extract larger-scale structure, what amounts to the musical score of the piece: the single waveform is recognized as the superposition of many separate waveforms emanating from distinct types of instruments, and each separated waveform is analyzed as a sequence of notes with approximately discrete pitches and durations. The notes of this multi-instrument score are "in" the sound wave in much the same way that individual words or concepts are "in" the neural state of a NECST model processing language: it takes some math to pull them out of the small-scale structure, but they are there, and they can be used for complex computation.

Beyond the score—and analogous to the phrase-, sentence-, and discourse-structure that can organize words at multiple levels in NECST language models—our listener perceives the still larger-scale structure of the movement: a *sonata-allegro* form in which two 'themes' (A, B) are introduced in the initial 'exposition' section, then repeated with variations in the second 'development' section (A′, B′, . . .), and then repeated in their original form in the final 'recapitulation' section. The large-scale themes and the *sonata-allegro* structure built from them are also "in" the sound wave, but a great deal of additional complex computation upon the score must be performed in our conductor's head to extract them.

Right: The levels of a NECST sentence encoding, from neural activation pattern to sentence structure. At the finest resolution, our sentence is encoded as a distributed pattern of continuous neural activation levels (black grid). To identify the global structure of this information encoding, called a *Tensor-Product Representation (TPR)*, we decompose this black pattern into the superposition (sum) of multiple patterns (only four shown here: lime, brown, yellow, violet). Each of these colored patterns is decomposed further; this decomposition is shown only for the violet pattern, which is decomposed into the superposition of two patterns, red and teal.

As shown in the insert at lower right (see explanation below), the red pattern is now broken down further, factored as the (tensor) product of two smaller patterns (shown as red strips): one encodes the prefix un, and the other encodes the Left-position role within an ordered-pair structure: [x —]. The red grid pattern thus encodes the partial structure [un —]; in technical terms, this is the *binding* of the *filler* un to the *role* Left-position, written un:*Left*. The other pattern (teal) is also decomposed as the sum of two patterns (green, blue), each of which is in turn factored as the tensor product of two smaller patterns: together, these reveal that the teal pattern encodes the substructure [lock able].

When all colored patterns are decomposed and factored in this way, we expose the global structure implicit in the original black pattern: the structure of an entire sentence. The patterns encoding the atomic fillers——un, lock, etc.——vary continuously depending on the context in which they appear. Furthermore, the patterns encoding *Left* and *Right* can also vary continuously, encoding more than just the discrete positions left and right: this produces a novel type of encoding in which *the compositional structure itself* is inherently continuous (see §2.2, p. 32).

Insert. The filler vector $\overrightarrow{\text{un}}$ (which encodes the symbol un), and the role vector $\overrightarrow{\textit{Left}}$ (which encodes the role *Left*) are bound together by the tensor product $\otimes$, producing the binding vector $\overrightarrow{\text{un}:\textit{Left}} = \overrightarrow{\text{un}} \otimes \overrightarrow{\textit{Left}}$. This type of vector is an order-2 tensor; its elements are visualized as forming a 2-D grid, each a number that is indexed by two subscripts specifying its row, column location. Each element is the product of two elements of $\overrightarrow{\text{un}}$ and $\overrightarrow{\textit{Left}}$. For example, in the 3rd row, 4th column position, there is $\overrightarrow{\text{un}:\textit{Left}}_{34} = \overrightarrow{\text{un}}_3 \times \overrightarrow{\textit{Left}}_4$.

A TPR can be embedded within a larger TPR; e.g., the vector that encodes [un [lock able]] is:

$$\begin{aligned}\overrightarrow{[\text{ un [lock able] }]} &= \overrightarrow{\text{un}} \otimes \overrightarrow{\textit{Left}} \oplus \overrightarrow{[\text{lock able}]} \otimes \overrightarrow{\textit{Right}} \\ &= \overrightarrow{\text{un}} \otimes \overrightarrow{\textit{Left}} \oplus [\overrightarrow{\text{lock}} \otimes \overrightarrow{\textit{Left}} \oplus \overrightarrow{\text{able}} \otimes \overrightarrow{\textit{Right}}] \otimes \overrightarrow{\textit{Right}} \\ &= \overrightarrow{\text{un}} \otimes \overrightarrow{\textit{Left}} \oplus \overrightarrow{\text{lock}} \otimes \overrightarrow{\textit{Left}} \otimes \overrightarrow{\textit{Right}} \oplus \overrightarrow{\text{able}} \otimes \overrightarrow{\textit{Right}} \otimes \overrightarrow{\textit{Right}}\end{aligned}$$

The role of lock in this embedded structure is *Left-of-Right*; this embedded role has encoding vector $\overrightarrow{\textit{Left}} \otimes \overrightarrow{\textit{Right}}$. The direct sum operation $\oplus$ is simply a generalization of vector addition that enables summing tensors of different order, which lie in different subspaces of the overall vector space. Binding operations other than the tensor product have also been used for neurocompositional modeling: see the first paragraph of §2 and note 5, p. 24.

Consider a simple type of symbol structure with only two positions: the ordered pair, exemplified by [lock able]; it is defined by two roles we can call *L* (left) and *R* (right). We can visualize *L* as $\left[\ \square\ —\ \right]$ and *R* as $\left[\ —\ \square\ \right]$. In [lock able], role *L* is filled by lock and role *R* is filled by able. This structure has two parts, each being a role filled by a particular symbol: each part is a *filler-role binding* which we write filler:*role*. Our pair can be written as lock:*L* & able:*R* and visualized as $\left[\ \boxed{\text{lock}}\ —\ \right]$ & $\left[\ —\ \boxed{\text{able}}\ \right]$: here two bindings are *aggregated* (by '&') to form a single structure. In NECST computing, this formal characterization of the structure is neurally encoded as a vector, as follows.

Each filler symbol is encoded by an activity vector, some list of real numbers (typically resulting from deep learning); this is commonplace in neural network modeling. A key innovation of NECST is that each *structural role* is similarly encoded by an activity vector. Then the two operations—':' for binding each filler to its role and '&' for combining the bindings—are realized as operations on vectors. The aggregation operation '&' becomes the direct summation of vectors: $\oplus$; this is represented by superimposing of patterns in Fig. 3. The binding operation ':' is a little more complex: it becomes the *tensor product* $\otimes$; this is shown in the inset at the lower right of Fig. 3.

To summarize, the symbol-pair [lock able] is formalized as a collection of filler-role bindings, lock:*L* & able:*R*, which in turn is encoded neurally as the vector arising by replacing each filler and each role by the vectors that encode them, and combining these vectors using two vector operations, one realizing ':' and the other '&'. The resulting vector is the ***Tensor Product Representation*** (**TPR**) that encodes the structure as a whole [198]. (This was also proposed in psychological memory models [166].) [Technically: $\overrightarrow{\text{[lock able]}} = \overrightarrow{\text{lock}} \otimes \overrightarrow{L} \oplus \overrightarrow{\text{able}} \otimes \overrightarrow{R}$, where $\overrightarrow{X}$ is the vector encoding of X.]

A technical consequence of using the tensor product to realize filler:*role* binding is that a TPR is a special type of vector—a *tensor*—in which the numerical elements are best visualized as arrayed in a multi-dimensional grid, rather than the one-dimensional sequence ordinarily used to display vectors; the dimensionality of the grid of values constituting a tensor is the *order* of the tensor. But the tensorial nature of TPR vectors will not matter for the non-technical presentation here.

Compositional encodings derive much of their power from compositional operations that allow one such encoding to be *embedded within* another [114]. Having composed lock and able together by respectively binding them to the *L* and *R* roles of the pair structure, we can now place *that* structure within a larger pair structure, by having it fill the role *R* of the larger pair: this is how we get [un [lock able]] (see Fig. 3). Crucially, the vector operations used to construct the TPR encodings of these structures can be embedded within one another. [Technically: $\overrightarrow{\text{[un [lock able]]}} = \overrightarrow{\text{un}} \otimes \overrightarrow{L} \oplus \left(\overrightarrow{\text{lock}} \otimes \overrightarrow{L} \oplus \overrightarrow{\text{able}} \otimes \overrightarrow{R} \right) \otimes \overrightarrow{R}$. The direct sum operation $\oplus$ is a version of vector addition that enables summation of tensors of different orders, which arise from different numbers of tensor products and lie in different subspaces of the overall vector space of all structural encodings.]

The notion of structural role is highly general, and subsumes the notion of *structural relation*. Some types of compositional structure are most conveniently defined in terms of structural relations between parts of the structure. A type of taxonomy structure, for example, can be defined by the relation is-a-kind-of ; e.g., this could relate poodle and dog in a taxonomy of English animal names. In symbol structures, structural relations, too, are discrete: an X either is-a-kind-of Y or it is not.

This is the complete description of the neural realization of compositional operations that are built into NECST computers. What about *de*composition operations, which take a structure and from it, extract its parts? These too are built into NECST computers, where decomposition means taking the TPR vector that encodes a structure—such as the pair [un [lock able]]—and computing from this vector the filler of some role, that is, *unbinding* the role: extracting the first part means unbinding the role *L* to get the vector that encodes un; extracting the second part means unbinding *R* to get the vector that encodes the smaller, embedded pair, [lock able]. The reason that the tensor product and direct summation are used for composition in NECST is that they naturally enable *de*composition. There is a simple vector operation that takes as input the TPR for a structure and the vector encoding of a role to unbind, and produces as output the TPR for the filler of that role in the input structure. [Technically, unbinding a role is taking the tensor inner product with the dual vector of the role encoding; see Fig. 9, p. 55].

Crucial to the power of symbol structures is that symbols maintain their identity as they play different roles [145, p. 1394]. The role *R* of [un lock] and the role *L* of [lock able] are filled by the same symbol, lock; in a symbolic language-processing model, this is why lock contributes the same meaning to both unlock and lockable—making compositional generalization possible. This critical property also holds for the continuous, explicitly

compositional vector encodings provided by TPRs, because the same vector that encodes lock appears in the vectors that encode the binding lock:*R* and the binding lock:*L*. TPR binding and unbinding operations preserve symbol identity: a filler can be bound to one role, then unbound and rebound to another role, then unbound again, all the while retaining its original encoding. The vector encodings of the bindings lock:*L* ([lock —]) and lock:*R* ([— lock]) disentangle, in a way that neural network operations can exploit, what is in common—the filler—from what is different—the roles.

2.1.2 Systematicity

The way TPRs disentangle fillers from roles—or what from where, or content from form—provides an opportunity for the processing of TPRs to deliver a strong form of compositional generalization. It is straightforward to constrain the strengths of the connections from a TPR encoding in one neural layer to a TPR encoding in another layer so that filler and roles are processed independently: this is the *systematicity constraint* [216]. Under this constraint, what is learned about how to process a filler when it is observed in one role immediately generalizes to processing that filler in any other role. And correspondingly, what is learned about processing a role when it is observed to be bound to one filler immediately generalizes to processing that role when it is bound to any other filler.

The intuition behind the systematicity constraint is illustrated by the translation task, where the English Adjective Phrase (AP) [black cat] becomes French [chat noir]. The fillers black, cat are respectively transformed to the fillers noir, chat, and the left- and right AP roles *AP-L*, *AP-R* are respectively transformed to *AP-R*, *AP-L*: they are reversed. To a first approximation, this translation mapping obeys the systematicity constraint, according to which the filler black is transformed to noir regardless of its role (e.g., adjective or noun) and the positional roles of the adjective and noun within the Adjective Phrase are reversed regardless of what particular words fill them.

Certain artificial tasks, like digit-copying, are so perfectly compositional that they exactly respect the systematicity constraint under an appropriate TPR encoding. But natural tasks, even those with a high degree of compositionality, will typically not observe the systematicity constraint perfectly. Natural languages exhibit many strong regularities, but unlike formal languages such as those designed for computer programming, most of these regularities have exceptions—so the systematicity constraint is not respected exactly. For example, the role transformation from English to French Adjective Phrases is not perfectly systematic: below we will see that little friend becomes petit ami—petit ('little') is one of a few exceptional adjectives in French that *precedes* the noun it modifies. Current work explores the design of networks that impose a systematicity *bias* which pushes learning towards processing that observes the systematicity constraint, but in a 'soft' manner that also enables exceptional non-systematic processing to the extent that the task requires it.

The remainder of this subsection (§2.1.3 – §2.1.4) presents a high-level summary of some of the evidence concerning the adequacy of NECST computing for providing neural computing with key elements of the compositional processing capabilities of symbolic computing.

2.1.3 Computability of symbolic functions

Do the capabilities built into NECST systems enable them to compute the kind of compositional functions needed for intelligent processing? Here 'function' has its mathematical sense of a transformation or mapping that takes an input and produces a specified output. Are the capabilities for creating and using compositional structures that are built into NECST models sufficient to enable neural networks to compute the powerful discrete functions for manipulating symbol structures that symbolic computing provides for intelligent information processing? These functions give traditional AI its strong compositional generalization ability. Since NECST-generation AI seeks to surpass the computational capabilities of traditional AI, we need to ensure that TPRs endow neural networks with at least the major aspects of the power of symbolic computing that is essential for AI.

Mathematical analysis has proved that, by encoding discrete symbol structures as TPRs, neural networks can in principle specify any Turing-computable function [203]; such analysis has shown explicitly how to compute classes of recursive functions that symbolic theories have shown to be important for human-level intelligence [206][7]. Even simple networks consisting only of an input layer and an output layer, each hosting the TPR encoding of a discrete structure, and simple linear connections from input to output, can compute functions like the one shown in Fig. 2, p. 20 (right panel) [205]: this model, LFNET, takes the phrase structure of an English passive-voice sentence such as Modern Artificial Intelligence was anticipated by Ada Lovelace and maps it to its meaning in logical form (LF): anticipated(Ada Lovelace, Modern Artificial Intelligence).

A more complex NECST network, TREEADJNET, can compute the Tree Adjoining function in Fig. 1*h*, p. 9, which takes as input the phrase-structure trees for the sentence Kim hates symbols and the adverb really and inserts the latter into the middle of the former to produce as output the phrase-structure tree for Kim really hates symbols. This is an operation that is known to give grammars sufficient power to achieve the level of complexity displayed by the syntax of sentences in human languages [91].

Regarding sentence meaning, a NECST network βREDNET can compute the composition operation that is fundamental to the formalization of the principle of compositionality in linguistic semantics, β-Reduction: applying a function defined over variables to particular values of those variables. A simple example from the domain of algebra, rather than sentence meaning, is shown in Fig. 1*j*, which applies the function $f(x) = x + (x/2)^2$ to the value $x = \sqrt{a}$. A NECST network INFNET can also perform basic logical reasoning, including transitive inference, illustrated in Fig. 1*k*–*l* [209].

2.1.4 Grammars emerging from neural computing

Can neurocompositional computing—in particular the NECST theory of how large-scale compositional-structure computing emerges from small-scale neural computing—contribute to our understanding of cognition at the large scale? A demanding test of any general framework for modeling cognition is its ability to advance our understanding of cognitive phenomena in which symbolic theory has made great strides but still faces fundamental challenges. One such arena, formally studied in depth since the 1960s

[7] NECST has thus made it possible, arguably for the first time, to explain by reduction to smaller-scale principles the systematic generative capacity of human cognition [201, 200], providing a resolution for an extended debate with Fodor and colleagues [125, 137] by correcting the influential but erroneous claim [53] that this capacity can be explained by theories of cognition based on symbolic, but not neural, computing. For a succinct resume of the debate through 2000, see [201, p. 513].

[31, 32, 42], is the space of possible grammars of human languages: formal linguistic theory. NECST has led to formal answers to the question: what types of grammar can emerge from neural computing [169, 202]? These new grammatical theories are a major departure from previous grammatical frameworks and have transformed parts of formal linguistics [61]. This provides evidence that NECST is a useful description of human linguistic intelligence.

The principle underlying the NECST approach to grammar is that encodings of linguistic structures are computed in networks with feedback loops in which activation from the input flows around continuously until it settles into a stable equilibrium pattern, the model's output (Fig. 11, p. 60). A class of networks of this sort compute encodings that maximize a numerical measure—*Harmony*—of the well-formedness of a neural state: co-activation of a pair of neurons raises (or lowers) Harmony whenever they are joined by an excitatory (or an inhibitory) connection [196]. The simple structure of TPRs allows the Harmony of the TPR of a discrete structure to be calculated directly from the symbolic description of the structure. The symbolic structures that maximize this Harmony are the grammatical ones.

Calculating the Harmony of a symbol structure involves summing the penalties incurred by that structure if it violates some of a set of proposed constraints that define a *Harmonic Grammar*, constraints such as 'a sentence must have a subject'; the magnitude of the penalty resulting from violating a constraint is the *strength* of the constraint in the grammar [113, 158]. A surprising empirical discovery is that for many grammatical phenomena, the strength of each constraint is greater than the sum of the strengths of all the weaker constraints, so that each constraint has absolute priority over all weaker constraints. In such a case, computing the grammatical structures requires only knowing the priority ranking of the constraints: the numerical strengths need not be consulted. This is a grammar of *Optimality Theory* [93, 112, 115, 170], which contributed to grammatical theory a fundamental new principle: the constraints of the grammars of all human languages are the same—grammars only vary in their priority ranking. This gives a formal means of calculating the set of all possible human languages predicted by a theory, by computing the properties following from all possible rankings of its proposed constraints [138].

It is possible to express standard symbolic probabilistic grammars as Harmonic Grammars and thereby encode them into feedback networks [63, 199]. Given such a grammar, a NECST network ParseNet can be designed in which the outputs are, to a good approximation, TPR encodings of discrete phrase-structures trees for grammatical sentences (see Fig. 11, p. 60). These sentence encodings are computed by continuous processing in continuous time over continuous distributed encodings in which each neuron simultaneously contributes to the encoding of all parts of the sentence [29]. Theoretical results show that in an idealized limit of unlimited processing time, these networks correctly output these grammatical structures with the correct probabilities [226]. Simulation results show that such convergence is often possible in psychologically plausible time [30]. These networks have been used to address empirically observed psychological and neurophysiological phenomena in human sentence comprehension [8] and production [15].

2.2 Neurally encoding continuous structure

How can non-discrete compositional structure be realized in continuous neural encodings? We now see that the NECST technique just presented for neurally encoding discrete compositional structure—TPRs—already provides encodings of continuous compositional structure as well. But why is continuous structure even useful? In this subsection, we discuss continuous structure present in cognition, and then the continuous structure invented by the early NECST models that have been constructed to date.

In neurocompositional computing, continuous structure is ubiquitous. The vectors that encode all discrete structures, such as [lock able], form a set of isolated points in the continuous vector space in which they lie: all the other vectors in the space encode continuous structures [208].

For an intuitive example of continuous compositional structure, consider a visual scene. In this type of structure, parts are linked by relations that are continuous. In a scene with a painting above a table, the painting and table are related by the above relation, which has a whole continuum of instantiations: directly-above-by-1.23m, and partially-above-by-0.8m-and-partially-to-the-right-by-0.24m, and so forth *ad infinitum*. Although spatial relations provide a concrete, intuitively clear case of continuous structural relations, in neurocompositional computing, continuous *abstract* structural relations are also crucial in domains in which structure has traditionally been considered to be fully discrete. One such domain is language, to which we now turn.

The simplest kind of continuous structure is a discrete structural type—say, the ordered-pair structure discussed above—in which a position is filled by a *partially-active* symbol. Whereas a role in a discrete structure can be filled with a vector encoding of a symbol, here the role is filled by a scaled-down version of that vector, in which all activation levels are multiplied by a value between 0 and 1. In the continuous ordered pair [0.1t a], the left role is filled by a very weak t, with activity level 0.1; the right role is filled by a full-strength a.

But continuous structure goes further than this. The vector that encodes the symbol t can be bound to the vector that encodes the discrete structural role L (visualized as $\left[\boxed{\text{t}}\ \text{—}\right]$), as it is in the encoding of the discrete structure [t a], or to R (visualized as $\left[\text{—}\ \boxed{\text{t}}\right]$), as in [a t]. But t can also be bound to the role that is encoded by the vector half-way between the vectors that encode L and R: this encodes a role we can visualize as $\left[\ \boxed{}\ \right]$. When the vector encoding t is bound to this continuous role, the result can be visualized as $\left[\ \boxed{\text{t}}\ \right]$: t fills a weak (half-strength) role that spans both the left and right halves of the pair. An equivalent visualization of the same encoding vector is $\left[\boxed{\text{t}}\boxed{\text{t}}\right]$ or [0.5t 0.5t]: L and R are both filled with a weak t, one that has half the activity level of a full, discrete t. [Technically: this continuous role's encoding is $0.5\overrightarrow{L} \oplus 0.5\overrightarrow{R}$, and binding the filler encoding $\overrightarrow{\text{t}}$ to it yields the TPR $\overrightarrow{\text{t}} \otimes (0.5\overrightarrow{L} \oplus 0.5\overrightarrow{R})$; this is the same as the vector $(0.5\overrightarrow{\text{t}})\otimes\overrightarrow{L} \oplus (0.5\overrightarrow{\text{t}})\otimes\overrightarrow{R}$ which is the TPR of a pair in which both the left and right positions are filled by a t with activity level 0.5.]

2.2.1 Continuous structure in linguistic cognition

How could an odd structure such as [t] be relevant to cognition? In fact, just such a continuous compositional structure has recently been proposed within theoretical linguistics to analyze a problematic phenomenon in the phonology of French: liaison [41]. The phrase written petit ami, which means 'boyfriend' (literally, 'little friend') is pronounced as a sequence of four consonant-vowel syllables: pe.ti.ta.mi. To which word does the second t belong? The spelling places it at the end of the first word, but the pronunciation groups it with the beginning of the second word. Having access only to pronunciation, young children often treat tami as one possible form for 'friend', and peti as one possible form for 'small' [26]—which adults do too, since, when not followed by a vowel, the word written petit is actually pronounced without a final t.

The proposed analysis [207] is precisely that t fills a weak role that spans the end of the first word and the beginning of the second; or equivalently, that the final position of the first word *and* the initial position of the second word are filled with a weak t, with respective activities 0.48 and 0.09, as revealed by a learning algorithm which shows how the location of the t starts at the beginning of the second word (as in children), and gradually shifts during learning to finally occupy the continuous role that spans the two words [152, 213]. This role can also be viewed as a (weighted) *role blend* of L and R.

This analysis of French liaison also illustrates another important aspect of continuous structure: *filler blending*. The word for friend actually needs to be (0.09 t+0.09 n+0.09 z)ami: its first position is a blend of consonants, containing not only a weak t, but also a weak n and a weak z. These appear in bon ami, bo.na.mi 'good friend' and les amis, le.za.mi 'the friends': just as petit has a final weak t, bon and les respectively have a final weak n and a final weak z.

The consonant blend in the initial position of ami is a *conjunctive* mixture, in which all elements are simultaneously present. This is a new type of structure in language analysis, quite unlike a superficially similar type previously used—a probabilistic mixture, which is *disjunctive*: it represents uncertainty about which one of the alternatives in the mixture is present—but certainty that only one is [13]. It is crucial that each time ami is pronounced, *all* its weak initial consonants are simultaneously present, so that the one that matches the final weak consonant of the preceding word (if there is one), will be selected for pronunciation.

The full analysis of French liaison shows that the proposed continuous structure can account for a wide range of data including both facts that have led some to favor a discrete word-final-t structure and facts that have led others to favor a discrete word-initial-t structure: neither discrete structure can explain all these data. In fact, in the past few years, continuous structures have resolved apparent paradoxes in a number of domains within theoretical linguistics: the relative weakness of certain consonants and vowels—'ghosts'—in a range of languages (like the final t of petit) [239, 213]; learning extremely complex systems of word forms [179]; the blending of two sentence structures in which a phrase simultaneously occupies two structural positions [16, 97]; the blending of grammars and expressions in bilingual speech, where phrases from two languages are composed within the same sentence [59]; and many other phenomena that have long defied satisfactory explanations within theoretical linguistics, which has previously been limited to purely discrete compositional structures.

Continuous compositional structure also provides a natural formalism for modeling human language *processing*. Consider the encoding, midway through a sentence, constructed by a listener having heard only Jay saw, in a discourse context containing two plausible candidates for who Jay saw: Kay and Dee [28]. A plausible encoding (highly simplified) is a continuous structure in which Jay fills the role of *subject*, saw fills the role of *verb*, and the role of *object* is filled with a vector encoding that is halfway between the encodings of Kay and Dee (an equal blend of the two). NECST neural models of human real-time sentence comprehension and production have shed fresh light on fundamental empirical patterns observed in laboratory psycholinguistic experiments [28, 29, 30] (see Fig. 11, p. 60).

These examples illustrate some of the ways in which the continuous compositional structure formalized in the NECST framework has strengthened the explanatory power of theoretical linguistics and psycholinguistics. This is evidence for our claim that the human cognitive computer uses neurocompositional computing—which is what we argue AI should emulate.

2.2.2 Learnability of continuous compositional structure in NECST models

How can neurocompositional computing incorporate deep learning to enable AI models to invent, and process, their own forms of continuous structure in order to optimally perform their task? We now present a few examples of how this has been done in early NECST-generation AI models; other examples are discussed in subsequent sections. First, however, we take up the question of whether existing networks can already learn compositionally structured encodings—specifically, TPRs—without having been endowed with special capabilities for doing so.

2.2.2.1 Spontaneously learned TPR structure. Recall the COPYNET-0 neural model tasked with accepting a sequence of digits as input, one at a time, encoding and storing the entire sequence, and then producing the same sequence as output, one digit at a time. One portion of the network takes in the input sequence and produces an encoding of that sequence; the other portion of the network takes in that encoding and then produces from it the output. When the network is trained on an unconstrained set of examples—a large randomly-chosen set of sequences—the resulting model performs perfectly on novel test items (unobserved during training). This is a successful case of compositional generalization. Could it be that this success results from the model having learned to produce compositionally-structured encodings? Perhaps even TPRs? Remarkably, it turns out the answer is yes: the learned encodings *are* TPRs (up to a linear transformation) [134]. So standard networks that lack explicit compositional-processing operations *do* learn TPR encodings—under ideal training conditions in which the network is exposed to a great variety of compositional configurations.

However, as mentioned in §1.1, p. 15 and shown in Fig. 4, p. 37 , when the training set is systematically constrained—e.g., excluding sequences with 1 in position *1*, 2 in position *2*, etc.—the result does *not* display compositional generalization.

The story is similar for a non-trivial, but still artificial, task: producing an appropriate sequence of basic commands for a robot to carry out a simple task described in highly simplified English [103]. This model, which we call ROBONET, displays compositional generalization when trained on unconstrained examples—and learns TPR encodings [216]. But when the training set is systematically constrained, compositional generalization fails.

Thus there is something so natural about TPRs for neurally encoding compositional structures that networks learn them spontaneously: but only if the training set provides overwhelming evidence for the compositionality of the task—which is possible in the simple, artificial tasks just discussed. However for the natural task of encoding sentences, even large training sets appear insufficient. There is only partial compositional generalization, and only partial evidence for learned TPR structure. The NECST method that revealed the covert TPRs in CopyNet and RoboNet also showed partial evidence of TPR structure in BERT [87].

Another indirect source of evidence that standard models learn representations sharing important properties with TPRs comes from 'analogies'. These were first proposed in psychology the 1970s [181] and rose to prominence in this century when reported in the highly influential neural model Word2Vec [140] that learned vector encodings of words: the vector difference between the encodings of king and queen is approximately equal to the difference between man and woman [141], a linear expression of the analogy 'king is to queen as man is to woman' ([124], but cf. [122, 165]). Such linear analogy relationships have been observed in many neural models since: they are now legendary, but their origin has been unclear. TPRs provide a principled explanation [52]: if words are encoded as structures with roles being semantic features such as social status and gender, the TPR relations are:

$$\begin{aligned}\overrightarrow{\text{king}} - \overrightarrow{\text{queen}} &= [\overrightarrow{\text{royal}} \otimes \overrightarrow{\text{status}} + \overrightarrow{\text{male}} \otimes \overrightarrow{\text{gender}}] \\ &\quad - [\overrightarrow{\text{royal}} \otimes \overrightarrow{\text{status}} + \overrightarrow{\text{female}} \otimes \overrightarrow{\text{gender}}] \\ &= (\overrightarrow{\text{male}} - \overrightarrow{\text{female}}) \otimes \overrightarrow{\text{gender}}\end{aligned}$$

Since the bindings to the status role cancel in the subtraction, this result is the same if we replace royal with commoner, giving $\overrightarrow{\text{man}} - \overrightarrow{\text{woman}}$.

Moving from words to sentences, the vector encodings of sentences produced by a recent mainstream model (Long-Short-Term Memory recurrent network, LSTM) [38] displayed analogies: e.g., the vector difference between the encodings of the two sentences the banker believed the professor and the judge believed the professor is approximately equal to the difference between the encodings of the banker helped the scientist and the judge helped the scientist: this must be true if the encodings are TPRs in which the corresponding words in each pair of sentences have the same fillers and roles, except for the replacement of the filler judge for the filler banker [134].

As mentioned previously, to learn robust compositional generalization without overwhelming evidence of the relevant compositional structure, we believe that neural networks need to be provided with special built-in capabilities for creating and using compositional structure: this provides them a bias that directs learning towards compositional analyses of the task, so that stronger compositional generalization can result from learning with smaller quantities of data. Whether such special capabilities are in fact *necessary* is, at this point, an open question.

What we now present is initial evidence that such capabilities, as provided by NECST computing, are *sufficient* to take AI in the direction of greater compositional generalization.

2.2.2.2 The NECSTransformer network . Recall that the structure built into one of the currently most powerful neural architectures, the Transformer, makes it a key example of 1G neurocompositional computing (§0.2, p. 11) [173]. We now present its upgrade to 2G neurocompositional computing, the NECSTransformer network.

A Transformer network [229] takes a sequence of symbols as input and produces a sequence of symbols as output; built into the network is a sequential structure of encodings that aligns with these symbol sequences. There are multiple neural populations or layers, each containing one sub-population for each symbol (in input or output); its activation pattern is often considered to be the encoding of that symbol in that layer. The encoding vector in a given layer for a given symbol collects a portion of its input activation from encoding vectors of other symbols in that layer. The network decides which other symbols a given symbol draws activation from—which it *attends to*—when processing a particular input symbol sequence. The input to one symbol's encoding from another's is multiplied by an *attention weight* computed by the network. Thus the flow of activation, or information, takes the form of a graph, where the link in the graph from one symbol to another has a continuous strength: the attention weight. The population of neurons that encodes a symbol (at a given layer) is subdivided into several subgroups, each with their own attention weights computed by their own attention 'head'. Thus there is a separate information-flow graph for each head (at each layer). We can think of each head as computing its own *structural relation* linking symbols: the greater the attention weight on information flow from one symbol to another, the more strongly the network deems those two symbols to be associated by this relation.

As noted above, in the Transformer language model BERT, some of these relations can be interpreted as approximating relations identified by linguists. For example, one is the co-reference relation that links two words referring to the same entity; in the input sequence She got some expert opinions on the damage to her home, this relates she and her. The head interpreted as computing this relation assigns a high attention weight for activation flow between the encodings of she and her, and relatively low attention weights for flow to these two encodings from the encodings of the other words [127]. Note that each head encodes the same relation throughout its layer, and the relations have no encodings of their own, hence no similarity to one another.

In the NECSTransformer, these relations are upgraded to first-class status with an encoding of their own: they serve as the roles defining a TPR compositional structure. For each head and layer, the network computes a relation-encoding vector for each symbol: these can vary across symbols, and the relation-encoding vectors have varying degrees of similarity to one another. For a given head, the attention-weighted activation flowing into a given symbol's encoding from other symbols serves as the filler for the relation computed by that symbol: this filler and role are bound by the NECST binding operation, the tensor product (or a compression of it); the binding vectors so produced at a given symbol—one per head—are assembled by the NECST combination operation, summation, to form a TPR structure that encodes a particular symbol in a given layer. The network uses deep learning to invent its own relations, distinguished by their encoding vectors; how these relations are filled is determined by the learned attention mechanism. Learning is driven to design structures that are optimal for performing the task.

Note that the TPRs invented by the NECSTransformer are *bona fide* information encodings: activity patterns passed on to other layers for further processing. In contrast, the graph structure implicit in a plain Transformer layer only transiently governs processing within that layer, and disappears after that processing terminates; it is not a stable

encoding that carries information to other parts of the network.

Once learning in the NECSTransformer is complete, it is up to the network's creators to try to interpret the learned structures: perhaps in terms of our pre-existing concepts about the task (such as co-reference), or perhaps by the modelers learning new concepts from the network's encodings. Interpretation of a NECSTransformer model for solving mathematics problems is discussed in the next section (§3.1, p. 40).

Returning to our expository example of the simple sequence-copying task discussed previously (§1.1, p. 15; §2.2.2.1, p. 34), we now ask: What happens when we upgrade the non-neurocompositional model CopyNet-0 to 2G-neurocompositional computing: a NECSTransformer network, CopyNet-2G? With this network's bias towards compositional encoding, an order of magnitude fewer examples of the copy task are needed to achieve the same degree of generalization as that of the plain (1G) Transformer model, CopyNet-1G (Fig. 4*a*–*b*). When multiple distinct instances of CopyNet-1G are separately trained on the task, only about 1/3 of them learn the task perfectly; upgrading to CopyNet-2G gives a 100% improvement (Fig. 4*d*).

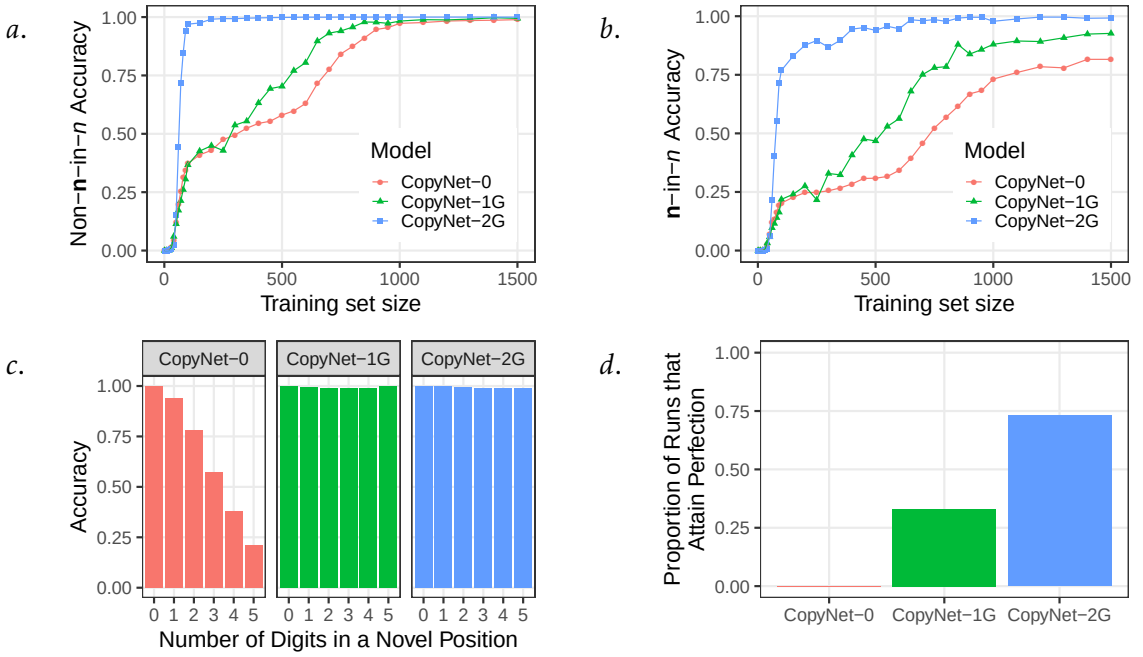

Figure 4: Successive increases in a model's degree of neurocompositional structure bring successive improvements in compositional generalization.

Figure 4: Neurocompositionality and compositional generalization

Increasingly deep implementation of neurocompositionality brings increasingly robust compositional generalization (for details, see Appendix B, §7, p. 57). This can be illustrated transparently by observing a sequence of successively more neurocompositional networks—called CopyNet-0, CopyNet-1G, and CopyNet-2G—that are trained to perform an extremely simple task: taking as input a sequence of 5 digits (e.g., ⟨3,9,7,4,7⟩), encoding the entire sequence as a vector representation, and then using that encoding to output the same sequence one digit at a time.

Compositional generalization is tested by withholding from training all sequences of a certain type: '*n*-in-*n*' sequences, in which there is a 1 in position *1*, or a 2 in position *2*, etc. After training, correctly copying the unseen *n*-in-*n* sequences requires a type of compositional generalization.

COPYNET-0 is a non-neurocompositional, pre-Transformer neural network which has little built-in neurocompositional structure (an LSTM [76]). COPYNET-1G is a Transformer, which uses a mildly neurocompositional graph structure (1G neurocompositional computing). COPYNET-2G is a more thoroughly neurocompositional NECST model (2G neurocompositional computing: a NECSTransformer, described in §2.2.2.2, 36).

a–b show how models with greater degrees of neurocompositionality perform better with smaller quantities of data. ***a*** gives accuracy on easier test examples which differ little from training examples, i.e., non-*n*-in-*n* sequences; note that COPYNET-2G achieves 100% correct with an order of magnitude fewer training examples. ***b*** gives accuracy on *n*-in-*n* test examples, which differ from the training examples and require more substantial compositional generalization.

c shows that COPYNET-0 does progressively worse on sequences with more and more digits in positions they never occupied during training, while COPYNET-1G and COPYNET-2G show little degradation in performance.

d. How do these neural learners' final performance levels compare to a symbolic learner, which would be expected to reach perfect accuracy (like COPYCOMP, §1.1.1, p. 16)? We trained each type of model 100 distinct times and counted how many of the 100 instances learned the task perfectly (scoring 100% on the set of all possible digit sequences of length 5—including all *n*-in-*n* sequences). No COPYNET-0 instances performed perfectly; about one third of COPYNET-1G instances reached perfect performance; and over two thirds of COPYNET-2G instances attained perfection.

2.2.2.3 Approximately-discrete structure in factual knowledge. Can NECST emulate an important property of the large-scale compositional structure of human cognition: approximately discrete compositionality? We now illustrate one approach to constructing such neural encodings during the inference of an output given an input. The idea is to first create the TPR vector that encodes a discrete structure appropriate for processing the input, and to then *improve* this encoding by sliding it continuously in its vector space to maximize some measure of the encoding's quality. This is optimizing the encodings of transient structures created during inference of outputs, building on the optimization, during learning, of the network's overall scheme (embedded in its connection strengths) for encoding the atomic concepts that combine to form these structures. This formalizes the contextual determination of meaning that concepts undergo when they are combined with other concepts to form a conceptual structure.

FACTNET is a NECST model of the storage and retrieval of factual knowledge. A useful discrete approximation of human factual knowledge, frequently employed in symbolic computational models, consists of a graph structure in which a set of symbols naming entities and properties are joined by links labelled by symbols naming relations. One fact in such a graph might be expressed by the symbol Barack_Obama being joined by a link

labelled office_title to the symbol US_President. FACTNET has *a priori* knowledge of the structure of this information: each of these three symbols is encoded by a learned vector, and the discrete approximation of the fact is encoded by a (compressed) TPR built from these vectors. The space containing these TPRs includes vector encodings of all assertions that can be built from the available symbols, e.g. "Hillary Clinton's office title was US President"; only some of these assertions are facts included within a given knowledge base. The set of these facts is thus encoded as a large collection of isolated vectors in the TPR space. This is discretely-formalized factual knowledge encoded within a continuous vector space.

The continuous character of this space is exploited by FACTNET to allow the encoding vector of any assertion to be optimized by being nudged in the encoding vector space away from its discrete approximation vector; this produces an encoding of a non-discrete, continuous structure. The model learns a validity function defined over the TPR space: it assigns high values to TPR vectors that encode assertions that are facts in a given knowledge base. These facts are the assertions that are, in a semantic sense, 'well formed'; they are analogous to grammatical linguistic structures, and the validity function is called Harmony, like its counterpart in the NECST-derived theories of grammar discussed above (§2.1.4, p. 30).

The optimized encoding of any assertion maximizes Harmony within the neighborhood of its discrete approximation. For each assertion, this has the effect of shading the encodings of all its constituent symbols to optimize validity within the context of the assertion they jointly define. So among the discrete approximation encodings of assertions of the form "*x*'s office title was US President", the two closest assertions to the assertion where x = Barack Obama are the assertions where x = Hillary Clinton and x = Al Gore; this is a consequence of the closeness of the encoding vectors—out of context—for Obama, Clinton and Gore. But, of course, these two closest assertions are not facts. After optimization, assertion encodings have shifted from their discrete approximations to improve their Harmony or validity scores, and now all of the five assertion encodings closest to the x = Obama case are assertions in which x actually was a US President. Out of context, Obama and Gore have similar encodings, but in the context of "office title", they do not. Evaluated on queries of a standard test knowledge base (WordNet18), the optimized continuous encodings of FACTNET set a new state of the art, raising the most stringent accuracy measure from 78% to 93%; optimized continuous encodings were essential to this improvement.

We have now outlined the NECST principles for achieving neurocompositional computing, illustrated how NECST AI systems are designed, and reported some of the improved performance that results. Finally, we pass to the matter of *understanding* NECST-generation AI systems, and exploring what benefits such understanding can bring.

3 Improved comprehensibility of neurocompositional systems and its benefits

Neurocompositonal computing imbues deep learning models with large-scale organization that emulates that of human cognition: encodings with approximately-discrete compositional structure. Does this better alignment with cognition begin to make the internal encodings of these systems less thoroughly opaque than those of typical deep neural models? This is of course an open question, as NECST-generation AI systems have only begun to be developed. Initial results, however, although fragmentary, do provide some grounds for optimism. For starters, as already discussed, the graph-structure building capabilities built into the 1G-neurocompositional Transformer architecture—which partially motivates building general compositional-structure capabilities into NECST systems—yield considerable insights when these networks' internal encodings are probed.

We now briefly describe early evidence of several benefits that enhanced comprehensibility can bring to 2G-neurocompositional NECST systems: learned internal encodings are more human-interpretable, more accessible for inserting structural knowledge that is expected to be valuable, and better for enabling human diagnosis of errors and control of system behavior. Such benefits have the potential for overcoming serious problems arising from the opaque nature of standard neural AI models, including difficulties in developing more explainable, trustworthy, and controllable AI systems [43].

3.1 Interpreting learned internal encodings

Interpreting the learned continuous compositional structure of NECST encodings can be illustrated with the MATHNET model [188]; this is a new version of the Transformer deploying 2G neurocompositional computing: the NECSTransformer, described above (§2.2.2.2, p. 36). This model learns to take in a sequence of characters spelling out a mathematics problem and to output a sequence of characters spelling out the answer; evidencing improved compositional generalization, the model succeeded in advancing the state of the art on this challenging task to 84.2% from the previously-reported 76% [187] (see Fig. 7*a*, p. 46 below). Recall that at each layer of such a network, each input symbol is represented by a TPR containing multiple bindings, where the roles and fillers are vectors generated within the network. The roles that MATHNET learns allow us to partially characterize the types of continuous structures it invents, as shown in Fig. 5. Each binding is computed by a different sub-network or head, and what is shown via color in Fig. 5*l*, for each of the input symbols, is the role generated by one head in the final layer of the input-encoding portion of the model. For interpretive purposes only, the role vectors generated by a head are grouped into discrete clusters, with similar role vectors in the same cluster interpreted to be encodings of essentially the same role; each cluster is assigned a color in visualizations such as Fig. 5*l*. Such a role can be interpreted by identifying the properties shared by symbols generating that role, and the properties relating those symbols to the symbols that provide the filler for that role. This enables interpretation of numerous roles invented by MATHNET.

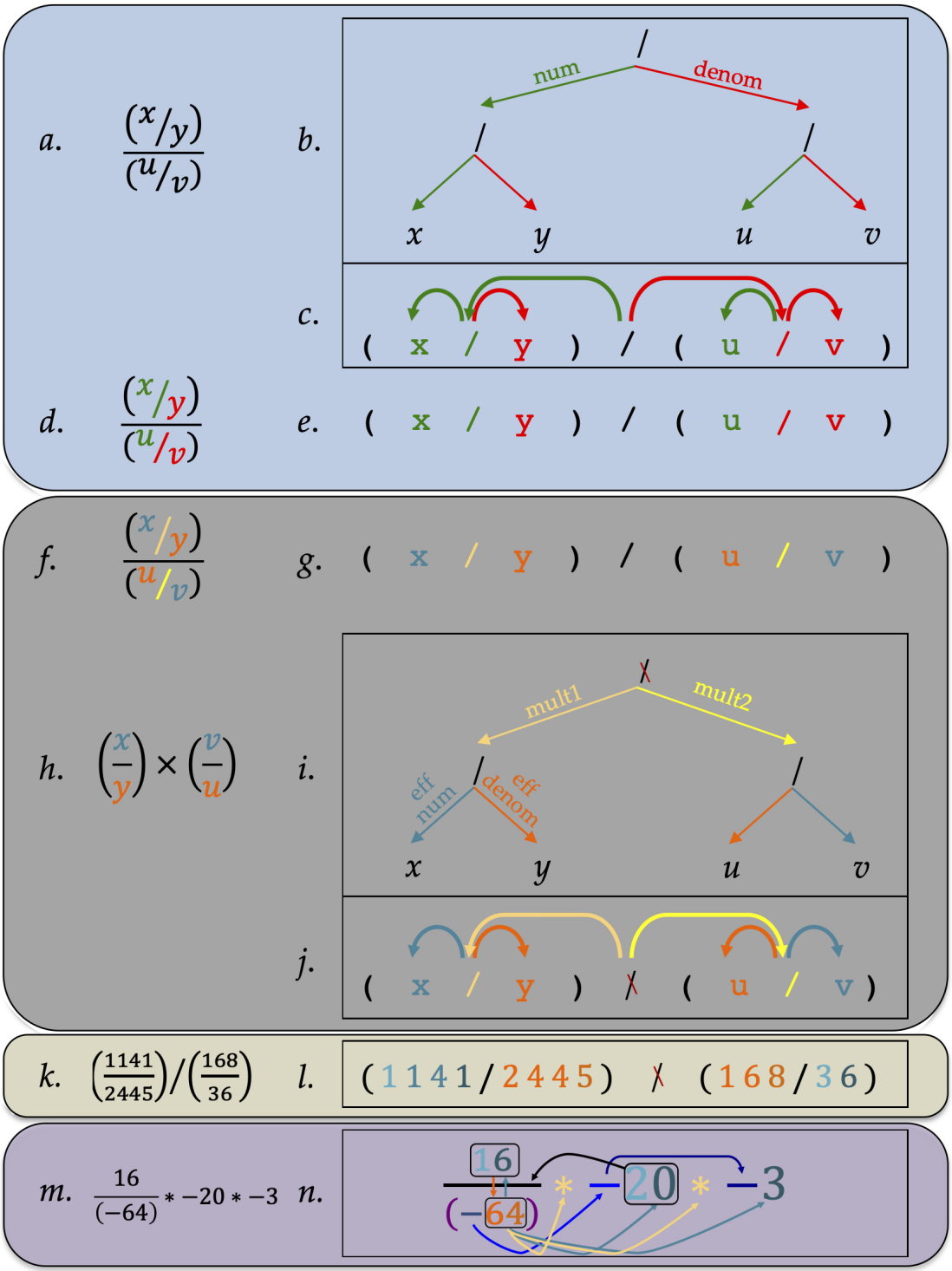

Figure 5: Decrypting the structures invented by MATHNET [187]

Figure 5: Decrypting the structures invented by MATHNET

Top: Standard structure. A ratio of ratios (***a***) can be represented as a tree structure (***b***). The topmost tree position (occupied by the central division operator '/') denotes the overall fraction, which is composed of two parts, occupying the two tree positions one level down. Each of these parts is also a ratio, again denoted by the division operator '/'. In the overall fraction, the leftmost ratio contributes the numerator role, marked by the green 'num' arrow; the rightmost ratio, the denominator role, marked by the red 'denom' arrow. Each of these two internal ratios in turn has two subparts, the first contributing the 'num' role, the second the 'denom' role. In the linear notation (which is what the MATHNET model sees), we denote the relations by arrows (***c***) and color each symbol by its role (***e***). Thus the colored version of ***a*** for this structure is ***d***.

Upper mid: Learned structure (simplified). But the structure invented by MATHNET is different: ***f***, or in linear form, ***g***. Now it is x and v, rather than x and u, that contribute the same role. The denominator of the denominator (v) fills the same role as the numerator of the numerator (x). The model's structural encoding reveals implicit use of the inference rule in Fig. 1*i*, p. 9: it has simplified the ratio of ratios into a product of ratios (***h***). Now the top-level operator, written '⁒', is the '/' symbol reinterpreted as the multiplication operator '×', with the two internal ratios now serving as multiplicands 1 ('mult1', gold arrow) and 2 ('mult2', yellow arrow). The invented roles shown here in teal and orange can be interpreted as effective-numerator ('eff num', including the denominator of a denominator) and effective-denominator ('eff denom', including the numerator of a denominator): the structure MATHNET has invented can be approximated as in ***i–j***.

Lower mid: The learned structure. For the ratio of ratios ***k***, the colors in ***l*** show the actual roles assigned by MATHNET. Effective numerator digits employ three roles, ranging from light to dark teal, marking leftmost, middle, and rightmost digits. Effective denominator digits use two roles, light to dark orange, the latter marking the rightmost digit.

Bottom: Additional structural roles learned. The structure MATHNET invents for the more complex structure ***m*** is shown in ***n***. Again, the teal and orange roles respectively mark three effective numerator factors and an effective denominator factor in a fraction; each of the former takes its filler from the latter and vice versa. (Fillers flow along the arrows.) The multiplication operators '∗' involve a different role (gold). The pair of roles generated by the first two negation operators '–' (lighter blues) correspond to negations that cancel each other, whereas the last, uncancelled, '–' employs a different role (dark blue). The purple role is one of two that mark matching parentheses.

Important note: These diagrams do not show the continuous nature of the actual invented structure: fillers actually result from combining contributions from multiple symbols, not a single one; the arrows have different numerical strengths; the fillers flowing along them are continuous vectors; and the roles themselves are continuous vectors that have been artificially made discrete, in order to make the visualization more human-interpretable, by clustering nearby vectors together and marking them with the same color.

There is one role that is typically generated by both an outermost opening parenthesis '(' and its matching closing parenthesis ')', but not the potentially many parentheses between them; e.g., it is generated by the first and last parenthesis of (((−10)/(−4))/1 + −1) but not by any of the six intervening parentheses. (In another problem, two such matching parentheses are successfully paired in this way, despite 31 intervening characters, including 10 parentheses.) Another role is typically assigned to plus signs '+' and also to each of two minus signs '–' that cancel each other, but not to uncancelled '–' signs. Other roles mark the position of digits within numerals that fall in the 'effective numerator' of a fraction, including numerators of simple fractions but also the denominator of a denominator in a complex fraction: at the layer in question, an expression like (3/4)/(5/6) is assigned the same structure as (3/4)∗(6/5), a ratio-of-ratios having been simplified to an equivalent product of ratios, in accord with the inference rule of Fig. 1*i*, p. 9.

Three other NECST models processing English invent structures that are partially interpretable in grammatical terms, although no information is given to the models about grammar or the structure of English. In a model that learns to answer questions about Wikipedia articles, QANET, the roles learned for TPR encodings of English questions partially align with recognizable linguistic structural properties, at levels ranging from small features of words (such as *plural*), to general semantic categories like *predicate* (expressed by verbs and adjectives) to an invented type of multi-word sequence that could be called a *wh-restrictor-phrase*, like what famous event in history. Another model, CAPTIONNET, learns to generate figure captions, in the process inventing roles that are highly predictive of the sequence of parts of speech (noun, adjective, etc.) in its captions [80, 81] (Fig. 8*b*, p. 48). In a NECSTransformer model that generates text summaries, SUMMARYNET, the invented structure parcels syntax into roles—and semantics into fillers [88].

Beyond the compositional structure within the representations of neurocompositional models, can we understand the *processing* within these models compositionally? (Recall the question raised in §1.1.3, p. 17 of whether the partial compositional structure that can be observed in some Transformer models plays a causal role in the processing within those models.) In MATHNET's evaluation of numerical expressions, the vector encodings of digits turn out to be sequentially arranged within the vector space according to their numerical value, from smallest to largest. This means that the semantic—numerical—value of a representation can be inferred from its position in vector space. Looking at a set of problems evaluating various numerical expressions with a given structure, we observe encodings consistent with a compositional process in which the innermost operators in the expression have representations corresponding to the numerical value resulting from applying the operator to its arguments, with values then propagating from the innermost to the outermost operators: see Fig. 6 [184]

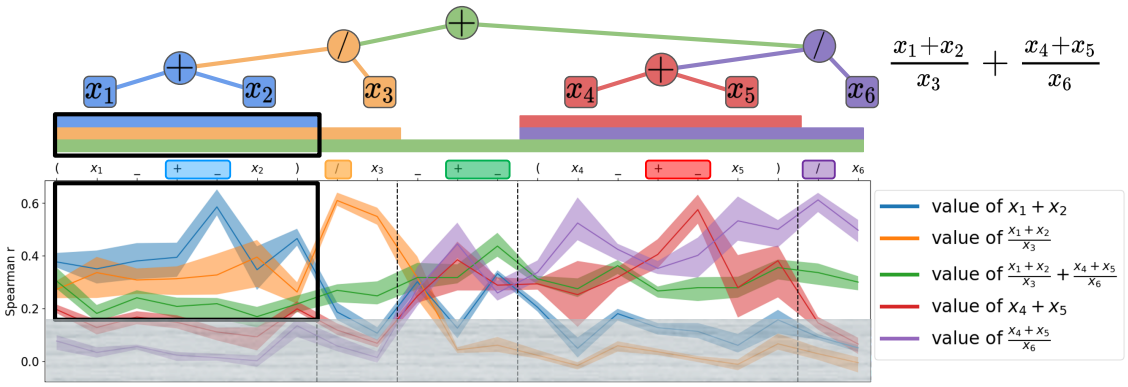

Figure 6: Compositionality of processing within MATHNET [184]

Figure 6: Compositionality of processing within MATHNET

Interpreting the encodings in the final layer of MATHNET [187], we see that the vector encodings of digits are sequentially arranged within the vector space according to their numerical value, from smallest to largest. The plot in the figure interprets the encodings of all symbols in expressions of the form $\frac{x_1+x_2}{x_3} + \frac{x_4+x_5}{x_6}$, where each x_i is a digit. Vertical dotted lines delineate sub-expressions, and underscores indicate space characters.

The vector encoding of the '+' symbol in $x_1 + x_2$ correlates with the numerical value of the sum of the values of the digits x_1 and x_2. (More precisely, the distance in vector space between the encoding of '+' when evaluating $x_1 + x_2$ and the encoding of '+' when evaluating $y_1 + y_2$ correlates well with the difference between the values of the two sums.) Similarly, the vector encoding of '/' in $\frac{x_1+x_2}{x_3}$ correlates with the numerical value of the ratio of the value of $x_1 + x_2$ to the value of x_3: the strength of such correlations are plotted by the curves in the figure. The value of sub-expressions propagates to the evaluation of the larger expressions of which they are a part: compositional processing.

The plot shows the alignment of correlation measures across the entire input sequence with the parse tree of the expression, which has been colored to match the lines in the plot for the appropriate quantities. (Colored envelopes show the minimum and maximum correlation across heads.) As shown by the colored rounded rectangles overlaying the expression, correlations with the values of an intermediate result are maximal at the vectors associated with the corresponding operator (or an adjacent blank '_'), and are elevated over the constituent containing its arguments (shown by matching colors in the parse tree, and matching colored horizontal bars covering the extent of the appropriate constituent). Where the correlations exceed 0.2 (above the grayed-out region at the bottom of the plot), the stacking of the correlation curves parallels remarkably well (although not perfectly) with the stacking of the colored bars marking the nesting of constituents.

An example is highlighted by the black boxes at the left: the vectors representing each element in the constituent $(x_1 + x_2)$ show the highest correlations with the value of the sum of x_1 and x_2 (blue), but also show correlations with its division by x_3 (orange), and the result of the whole expression (green).

3.2 Inserting structural knowledge: Biasing structure learning

Can the improved interpretability of neurocompositional computing enable us to endow a model with structural knowledge that we expect to be useful for its task? NECST-generation AI promises many opportunities of this sort. The method we illustrate here biases networks to create TPRs of a certain type of structure, pushing deep learning towards encodings of the task that use this structure.

The NECST model PROGNET (Fig. 7*c*) accepts a statement of a problem in English and outputs a program that solves the problem: the program takes the form of a sequence of commands, each consisting of an operator (such as 'plus', or 'sort') together with 1–3 entities (such as numbers or symbol-lists) that it operates on—its *arguments*. What is given to the model is knowledge of the structure of the output: a list of operator-argument commands; previous models typically generate output programs symbol-by-symbol, oblivious to any larger-scale structure.

PROGNET has been applied to two cases: mathematics word problems and programming problems in the Lisp programming language (modified to consist in a sequence of commands). The input problem is encoded by a network component that takes in one word at a time, assigning each a vector that encodes a filler:*role* binding, where the filler- and role-encoding vectors are each selected from a learned dictionary of such vectors. (No prior knowledge relevant to understanding English is provided.) These individual-word binding vectors are aggregated to form a single TPR that encodes the entire problem statement.

This TPR is transformed by a second model component into an encoding that drives the entire output-generation process, which is executed by a third model component. This third generation component produces a sequence of *command encodings*, each of which it uses to generate a program command, using TPR unbinding, as follows. Each command encoding is assumed to take the form of a TPR for a 4-tuple: there are four roles, one for the operator, the others for its arguments. The generator uses TPR unbinding to extract the fillers of these roles from the command encoding, outputting for each such extracted filler encoding the closest symbol encoding in a learned dictionary of encodings of all possible operator and argument symbols.

This model uses TPR binding to encode the input problem (using the method of QANET [157]), and TPR unbinding to generate the output commands (using the method of CAPTIONNET [81]). The built-in compositional-structure processing capabilities produce an input TPR, by construction; but the unbinding capability's job is more subtle. The command encoding is not forced by construction to be a TPR; it is produced by standard network operations which themselves impose no structure on the encoding. However this encoding is processed by TPR unbinding, so to succeed at its task, the model must learn to create command encodings that are TPRs, as far as the command-generator can see. In this way, the learning process is biased to produce command encodings with TPR structure: it is via this bias that useful structural knowledge is provided to PROGNET.

With this learning bias, PROGNET's performance set a new state of the art for both the mathematics and programming problem-solving tasks. Lisp programs as long as 55 commands were generated perfectly. The use of TPR structure for encoding the English input, and the bias for TPR structure in the output-generation process, were both essential for the model's strong performance (Fig. 7*c*).

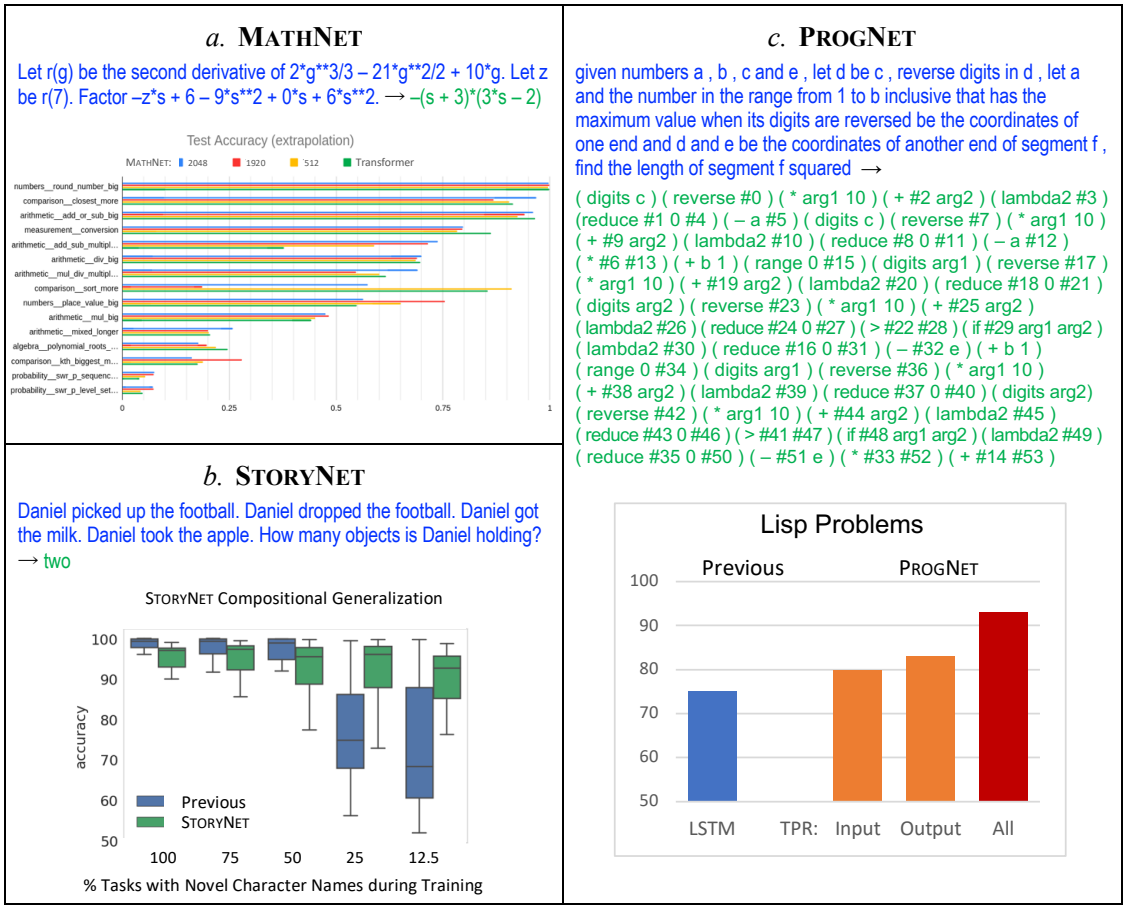

Figure 7: NECST-generation AI models—Improved compositional generalization: MATHNET [187], STORYNET [186], PROGNET [25]

Figure 7: NECST-generation AI models—Improved compositional generalization

a. An example problem (in blue) to which MATHNET [187] gives the correct answer (in green). Performance on the 15 particularly challenging test sets that require most combinatorial generalization away from the training examples are shown in the plot. For 11 of these test sets, the plain (1G) Transformer (green bars) is surpassed by the (2G) NECSTransformer model, MATHNET (shown in three model sizes: yellow, red and blue, in increasing order).

b. An example narrative with a question correctly answered by STORYNET [186]. Compositional generalization was tested by restricting training examples that included certain character names to only some fraction of the multiple subtasks in this dataset. A name that was included in all subtasks is plotted on the left; one that was included in only 12.5% is shown on the right. Green shows the level of generalization on the 'novel' names by STORYNET; blue, the performance by a previous state-of-the-art model. As the percentage of subtasks in which each 'novel' name was included in training decreases, STORYNET increasingly dominates.

c. An example Lisp programming problem that PROGNET [25], but not the previ-

ous state-of-the-art (LSTM) model, answers correctly (matching all 55 commands exactly). ('#*n*' refers to the result of the n^{th} command.) The plot compares the overall performance of the LSTM model to versions of PROGNET in which TPRs were used for only the input, only the output, or both.

Another NECST model that was provided with valuable learning biases for compositional learning was a model for answering questions about short narratives like the one in Fig. 7*b* [232]: STORYNET [186]. This network learned to encode abstractions of entities and relations as vectors and to use built-in operations to bind them together into a TPR encoding of a continuous graph structure capturing knowledge of the events in the narrative. The learning bias provided to this network was a set of built-in useful operations for updating the knowledge graph as each sentence of the narrative arrives, and for sequentially extracting information from the graph to answer a question. The model successfully learned how to use these operations. Not only did this model advance the state of the art on its task, it exhibited dramatically improved compositional generalization when tested with novel words on which it had received quite limited additional training (Fig. 7*b*).

3.3 Diagnosing errors and controlling output

Is the comprehensibility of NECST AI models sufficient to enable us to identify internal processing errors and to intervene on the internal encodings to exert control of the models' outputs? Initial results show that the interpretability of the fillers and roles in internal TPR encodings can sometimes reveal where the model selected the wrong filler or role, and allow us to control model output by altering the fillers and roles directly.

3.3.1 Diagnosing errors

Recall that the NECST model QANET, which answers questions about a set of Wikipedia articles, invents a number of roles that can be interpreted grammatically (§3.1, p. 40). In addition, some fillers can be interpreted semantically (see Fig 8*c*). The word Who has several meanings, and which filler is assigned to a particular instance of Who correlates well with which meaning is appropriate for the specific context of that instance. In the many mentions of the TV character Dr. Who in the articles, the model sometimes misassigns Who the filler that would be appropriate for a question word, as in Who died? (although QANET made this error at a considerably lower rate than a state-of-the-art probabilistic symbolic system designed for analyzing English sentences [126]). When the model made this internal "error", it was 5 times more likely to produce an incorrect output: an output that would have been appropriate had Who actually been a question word rather than a character name [157].

3.3.2 Controlling output by manipulating learned compositional encodings

The incomprehensibility of contemporary black-box non-compositional neural AI models severely limits our ability to control the socially toxic biases these models often display [10]. The relative comprehensibility of NECST encoding vectors—TPRs of continuous

compositional structures that can often be approximated by symbol structures—can, however, enable 'precision surgery' on internal activation patterns to control the output of a network, something that is very difficult with traditional non-compositional neural models (but cf. [58]; related operations are also possible with other 1G neurocompositional Transformer models [118]).

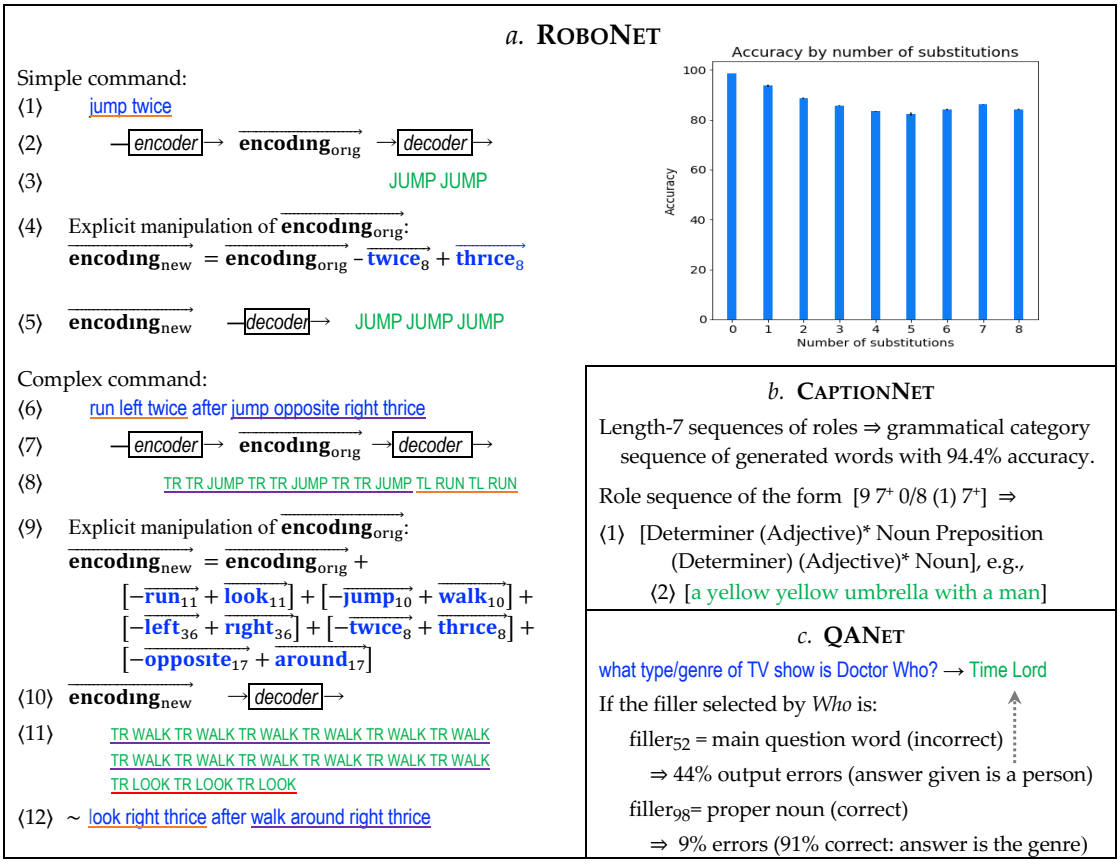

Figure 8: NECST-generation AI models—Benefits from increased comprehensibility: controllability, ROBONET [216]; interpreting internal dynamics, CAPTIONNET [81]; error diagnosis, QANET [157].

Figure 8: New capabilities resulting from increased comprehensibility of neurocompositional computing

***a:* Controlling behavior** [216]. In this task, the input ⟨1⟩ is a description in simplified English of a maneuver for a hypothetical robot; the output ⟨3⟩ is a sequence of primitive operations the robot can perform to carry out the maneuver. (TL/TLdenote 'turn left/right' by 90°.) Depicted schematically in ⟨2⟩, this model, ROBONET, is a standard, non-neurocompositional neural network (Gated Recurrent Unit network, GRU [27]) in which an encoder subnetwork takes the input in, one symbol at a time, and produces a single vector ($\overrightarrow{\textbf{encoding}}_{\text{orig}}$) that encodes the entire input; this vector is then fed to a decoder subnetwork to produce the output, one symbol at a time. NECST analysis methods show that ROBONET's learned

encodings are well approximated by TPRs, yielding a closed-form expression for the learned vector ⟨2⟩ that internally encodes any given input ⟨1⟩. This expression tells us exactly how to take the internal encoding of an input such as jump twice and directly modify it into the encoding that would be produced by the input jump thrice: we subtract the vector hidden in this encoding that encodes twice and add the vector that would encode thrice ⟨4⟩—in the process, changing the activation level of every neuron by a precisely-determined amount. Now the output of the model changes from JUMP JUMP ⟨3⟩ to JUMP JUMP JUMP ⟨5⟩; that is, the alterations that we have made to the model's internal encodings produce exactly the output alteration we intended to make, illustrating how the interpretable structure of TPRs facilitates control of a model's behavior.

As shown in ⟨6 – 12⟩, this procedure can be repeated to effect considerably more complex modifications, changing the internal encoding ⟨7⟩ of the eight-word input ⟨6⟩ to what would be the encoding of the virtual input ⟨12⟩, which differs in five words. The eight-word input in ⟨6⟩ produces the encoding vector ⟨7⟩ and output ⟨8⟩. We can change this encoding to match that of a virtual input that differs from it in five words ⟨12⟩ by taking five steps of subtracting the encoding of an existing word and adding in the encoding of a new virtual word ⟨9⟩. ($\overrightarrow{\text{run}_{11}}$ is the vector encoding of run:$role_{11}$, the binding of filler run to role $role_{11}$. The analysis technique determines the role that run fills——$role_{11}$——as well as the vector that encodes this role, $\overrightarrow{role_{11}}$, and the vector $\overrightarrow{\text{run}}$ that encodes its filler, run; thus we can explicitly compute this contribution: $\overrightarrow{\text{run}_{11}} = \overrightarrow{\text{run}} \otimes \overrightarrow{role_{11}}$). The resulting encoding, when fed to the decoder ⟨10⟩, produces the output ⟨11⟩ that correctly corresponds to the virtual new input ⟨12⟩: the original 13 output commands have been replaced by 30 new ones. The plot shows how, even after many such successive substitutions, the internal modifications specified by our NECST analysis continue to control behavior with good accuracy.

b: **High-level interpretation of network-internal dynamics** [81]. Given an input image, the NECST model CAPTIONNET generates an appropriate caption one word at a time by first computing an encoding for the entire caption, which it then treats as a TPR: it generates one role at a time and extracts from the encoding the filler of that role, which is the word it produces. The roles it invents carry significant information about the grammatical category (part of speech) of the words it generates, such that if we know that it selected a particular sequence of 7 roles to generate 7 words then we can predict the grammatical categories of those 7 words with 94.4% accuracy. It learns to produce recurring sequences of roles which function like subprograms. One example is the pattern [9 7^+ 0/8 (1) 7^+] where '7^+' means one or more repetitions of role 7, '0/8' means either role 0 or role 8, and '(1)' means one or zero instances of role 1. This subprogram generates words forming the sequence of grammatical categories given in ⟨1⟩, where '(Adjective)*' denotes a sequence of zero or more adjectives. An example word sequence this produced is given in ⟨2⟩.

c: **Diagnosing errors** [157]. Given a Wikipedia article and a question about it, the NECST model QANET produces an answer. It encodes each word as a simple TPR consisting of a single filler:*role* binding, choosing from learned dictionaries of

20 possible roles and 100 possible filler symbols: none of these has any meaning prior to learning. After learning, several fillers can be interpreted as carrying identifiable meaning, including fillers 52 and 98, which respectively correspond to the meanings of Who as either a main question word or a part of a name (Dr. Who, that is). For the example question shown in blue, the model gave the incorrect answer in green. We can explain this error by looking at the network's internal encodings of the words in the question: we see that Who was encoded with filler 52, and the answer reflects this error, as the answer is a character name, which would be an appropriate type of response if Who were indeed the main question word. We can validate this explanation by observing that, on questions about Dr. Who, the model's error rate was 44% when it chose filler 52, but when Who was correctly encoded as filler 98, the error rate was 5 times lower: 9%.

This surgery is illustrated by ROBONET, the previously-introduced model that learns to map a simplified-English description of a robot maneuver into a sequence of primitive commands executable by the robot [216] (§2.2.2.1, p. 34; see Fig. 8*a*). As discussed, although ROBONET has no built-in capabilities for using TPRs, when trained on extensive, unconstrained examples, the internal encoding of the input that it learns is closely approximated by a specified TPR, revealing directly its otherwise covert compositional structure and providing an explicit formula expressing, in closed form, the vector that internally encodes any input. In the simplest type of example (Fig. 8*a*⟨1 – 5⟩), although every neuron simultaneously contributes to the encoding of all input words, the network's internal vector encoding of the input jump twice can be directly altered, changing the activation of every neuron by a precise amount, in such a way as to subtract the vector contribution to the TPR from twice and add the vector that would have been contributed by thrice—thereby changing the model's output from JUMP JUMP to JUMP JUMP JUMP. Much more complex changes, and extended sequences of successive changes, can also be successfully effected in this way (Fig. 8*a*⟨6 – 12⟩).

4 Summary

The long-anticipated arrival of Artificial Intelligence is surely one of the singular milestones of our time, and it seems an appropriate moment to reflect upon the field: its current status, where it's been, and where it's going. At the beginning of the computational era in the mid-20th century, modern AI and cognitive science were born together and developed hand-in-hand for several decades before gradually diverging [37]. We suggest it is time for the study of intelligence in humans and in machines to join forces once again.

The analysis of AI we have offered in this tutorial is the product of a cognitive science perspective on intelligence. We now summarize major symptoms of malaise that we perceive in current AI, our diagnosis, our treatment plan, and some early outcome assessments.

Among the symptoms of current difficulties are a failure to satisfactorily achieve human-like generalization, the pursuit of which has given AI an insatiable appetite for more learning data, for bigger models, and for more computing power to make it possible

to train these models on these data—exacerbating already pressing societal problems [10, 159].

Recent progress has already led to genuinely useful AI systems in myriad application areas: computing will never be the same. But while performance of AI systems has dramatically improved, how current systems actually perform their amazing feats is now a great new mystery.

Further, despite all this progress, there remains a large gap between machine and human intelligence. This leaves plenty of room for skeptics to say, "we've heard over-hyped stories about AI for 70 years; why should we believe that things are different now?". The history of AI has indeed been infamous for its multiple cycles of boom-and-bust.[8] Traditional AI blossomed for decades and then collapsed in the 1980s: the first "AI winter" [40, p. 203]. Artificial neural network AI systems then flourished, only to see the same fate in the 1990s. Is it inevitable that the current imposing edifice of neural-network AI will come crashing down as well?

Our diagnosis is that traditional AI and neural network AI—both drawing heavily from cognitive science—each got the story half right. Rather than continuing the pendulum swing between these two types of computing, we can stop the oscillation by properly understanding the type of computing that constitutes human cognition. The human mind/brain has organization at multiple scales. At a small scale, it is a neural network: the brain, with information encoded in numbers—neural activity levels. At a large scale, it has quite a different organization: the mind encodes information in intricate compositional structures. Traditional AI got right the large-scale compositional structure, but went wrong in formalizing this as discrete symbol structures. These have proved too rigid for adequately encoding the subtleties of information in non-formal intelligence-requiring tasks, and intractable to use because of the explosion of possible structures that must be sifted to find the correct one. Neural AI got right the small-scale structure: continuous, optimized numerical encodings supporting statistical inference, but went wrong in having no appropriate large-scale compositional structure. This omission has led to the inadequate compositional generalization that plagues contemporary AI systems. And the missing large-scale structure means the lack of the kind of anchors onto which human concepts can be attached to build understanding.

The treatment then is clear: design AI systems in which large-scale compositional structures are built not of discrete symbols, but of continuous neural activation vectors. This is neurocompositional computing. Having acquired a highly limited degree of neurocompositional computing as a side-effect of pursuing quite different goals, the Transformer architecture has led to great breakthroughs in language processing, and increasingly in many other areas of AI too—e.g., in mathematics: MATHNET (§3.1, p. 40) and INTNET (§1.1.1, p. 16). A way to take this progress much further, and achieve general, fully-neurocompositional computing is provided by Neurally-Encoded Compositionally-Structured Tensor computing: NECST (§2, p. 24; Appendix A, §6, p. 55).

Although it is certainly much too early for definitive assessment of the efficacy of this treatment, the accomplishments of the earliest examples of NECST-generation AI systems are encouraging. The results we presented are summarized in (6); these include theoretical

[8] Interestingly, such a cycle was also anticipated by Lovelace: "It is desirable to guard against the possibility of exaggerated ideas that might arise as to the powers of the Analytical Engine. In considering any new subject, there is frequently a tendency, first, to overrate what we find to be already interesting or remarkable; and, secondly, by a sort of natural reaction, to under value the true state of the case, when we do discover that our notions have surpassed those that were really tenable" [14, Note G., p. 398].

results (6*a*–*b*), network analysis methods (6*e*), and, as illustrations, 5 hand-designed (6*a*) and 7 deep-learning (6*b*–*d*) NECST models. In (6), '*' marks NECST models that set a new state-of-the-art level of performance.

(6) **Summary of results presented in this tutorial**

a. Discrete symbolic compositional functions known to provide important approximations to cognitive functions targeted by AI tasks are provably computable by precisely-designed neural networks (including LFNET, TREEADJNET, βREDNET, INFNET) using computing that is fully continuous, parallel, and distributed [205, 206] (§2.1.3, p. 30; Fig. 9, p. 55). Arbitrarily close approximations to discrete grammatical structures can be computed by neural networks that are fully interpretable, continuous, and distributed; these networks are formally derived from general theoretical principles, and can be used to model neural [8] and behavioral data on real-time human language processing (PARSENET [29] §2.1.4, p. 30; Fig. 11, p. 60).

b. New theories of grammar (Harmonic Grammar, Optimality Theory) can be derived from formally characterizing the compositional structure emerging from language-processing neural networks; these theories have led to significant progress in linguistic theory [158, 170] (§2.1.4, p. 30). Explanations in theoretical linguistics and psycholinguistics can be strengthened using encodings with continuous compositional structure [179, 213] (§2.2.1, p. 33). For answering questions through inference from a factual knowledge base, approximately-discrete transient encodings, partially interpretable as contextualized encodings of concepts, can be constructed during inference (FACTNET* [105], §2.2.2.3, p. 38).

c. Novel, partially-interpretable continuous structured internal encodings optimized for mathematics problem-solving are invented by the NECSTransformer enhancement of the original Transformer network architecture (MATHNET* [187], Figs. 5–6, p. 41). Continuous structured internal encodings learned for comprehension and production of English are partially interpretable in terms of grammatical concepts (QANET [157], CAPTIONNET* [81], SUMMARYNET* [88], §3.1, p. 40; Fig. 8*b*, p. 48). Such interpretation enables certain errors in the output of a question-answering model to be diagnosed as due to incorrect use of its learned internal vector encodings (QANET [157], §3.3.1, p. 47; Fig. 8*c*).

d. Neural networks can be informed by directing their learning towards useful types of continuously-structured encodings for generating computer programs to solve math and programming problems (PROGNET* [25], §3.2, p. 45; Fig. 7*c*, p. 46). Dramatically improved compositional generalization by neural networks answering questions about narratives can be achieved through providing them a novel framework for processing events by encoding them with continuous compositional structure (STORYNET* [186], §3.2, p. 45; Fig. 7*b*, p. 46).

e. For any given input to a neural network generating simplified robot control sequences, the output can be controlled by directly modifying the model's learned internal vector encoding of the input (ROBONET [216], §3.3.2, p. 47; Fig. 8*a*). The precise alterations necessary are derived from a closed-form formula for this vector, which results from novel analysis methods for uncovering latent compositional structure in neural encodings [134] (§2.2.2.1, p. 34).

5 Towards full neurocompositionality

The NECST work to date is only a first step towards fully achieving neurocompositional computing; to imbue networks with greater capabilities for processing continuous compositional encodings, several further steps are clear, mapping out a road to **3G neurocompositional computing**.

The early work within the NECST paradigm reported here has incorporated only the most fundamental features of neurocompositional structure processing (those identified in Fig. 9*a–j* of Appendix A, p. 55). This has just begun to manifest the power inherent in symbolic computing: it has improved compositional generalization, advancing the state of the art in several problem domains where compositional structure processing is particularly important, and it has improved interpretability of internal representations. The gains, however, are clearly still short of those expected from full achievement of human-level neurocompositional computing.

Further progress is expected from endowing deep learning with three additional capabilities of compositional-structure computing. The first is the capability provided by TPRs to explicitly embed smaller structures within larger structures, such as how [lock able] can be embedded within [un [lock able]] (Fig. 3, p. 25 and Fig. 9*k–l*, p. 55). Initial results in this direction have yielded promising results.

The second additional capability pertains to the connection weights that process the structured representations in these networks (Fig. 9*m*). NECST representations disentangle the filler ('what') and role ('where') aspects of structural constituents, but to date there is no explicit bias in NECST models that leads network processing to exploit this disentanglement. Building such a bias into the network structure will be a key component of 3G neurocompositional computing. A strong form of compositional generalization—pure systematicity—arises when fillers and roles are processed independently, as exemplified above by the transformation of English black cat to French chat noir (§2.1.2, p. 29 and Fig. 9*n*). Currently under development are NECST models that bias learning in this direction by favoring weight matrices that factor as the tensor product of weights that process fillers and weights that, independently, process roles.

The third new capability is the most complex. The transformative advances in grammatical theory derived from NECST center on the innovation that grammatical expressions are encoded in activation vectors that optimally satisfy grammatical constraints that are encoded in the strengths of connections joining neurons [169, 170]. Incorporating such optimization-based processing into 3G neurocompositional computing may set the stage for major breakthroughs in machine language processing [152], as it did in linguistic theory. An initial step in this direction, discussed above (§2.2.2.3, p. 38), shows how NECST can strengthen inference ability by enabling the meaning of symbols denoting particular entities and relations to adapt continuously to—to be optimized for—their structural context within a knowledge graph of facts (FACTNET, [105]). This just begins to show how continuous NECST compositional representations can show greater flexibility than their discrete symbolic counterparts.

The NECST generation of AI models promises to enable us to overcome the limitations of discrete symbolic implementations of compositional-structure computing while allowing us to interpret internal continuous representations, alter them in controllable

ways, and explain their consequences. NECST computing also offers the potential to instill in AI models information that is known from symbolic theories to be generally useful, including knowledge that is purportedly innate in human infants, greatly reducing the quantity of experience needed to achieve the levels of cognitive competence observed in human adults [129, 133]. The NECST paradigm, perhaps, may even allow us to gain new insights into specific cognitive capacities from the structures invented by these models to solve intellectually-challenging problems.

Cognitive science and AI emerged in close contact in the 1960s [37], but have since gone their separate ways. The research program reviewed here pursues a reconvergence of AI and cognitive science through a unified theory of the computing underlying both human and machine intelligence. Recognizing the importance of such a synergy is not new: it was already envisioned nearly two centuries ago by Ada Lovelace, who saw that in the general purpose computer, "not only the mental and the material, but the theoretical and the practical, are brought into more intimate and effective connexion with each other." [14, Note A, p. 369].

So why was Lovelace skeptical about the possibility of creativity in machines? "The Analytical Engine has no pretensions whatever to *originate* anything. It can do whatever we know how to order it to perform." [14, Note G, p. 398]. Regarding a deep-learning network, it may be true that we do not "know how to order it to perform" creatively: but what is new is that *we don't have to*—because the capabilities of the network emerge from its statistical analysis of its experience: we don't program these capabilities into networks in the sense that Lovelace knew from her programming of the Analytical Engine, nor in the sense that traditional AI systems were programmed. In deep-learning systems, large-scale organization arises from small-scale learning processes: it is not supplied by a programmer. AI systems that deploy neurocompositional computing have the capability to emulate both the small- and large-scale structure of human cognition; perhaps small-scale learning processes operating under strong compositional inductive biases will ultimately enable the emergence of the large-scale creative abilities characteristic of human cognition.

6 Appendix A. Compositional-structure processing: Formalizations for symbolic and neural computing

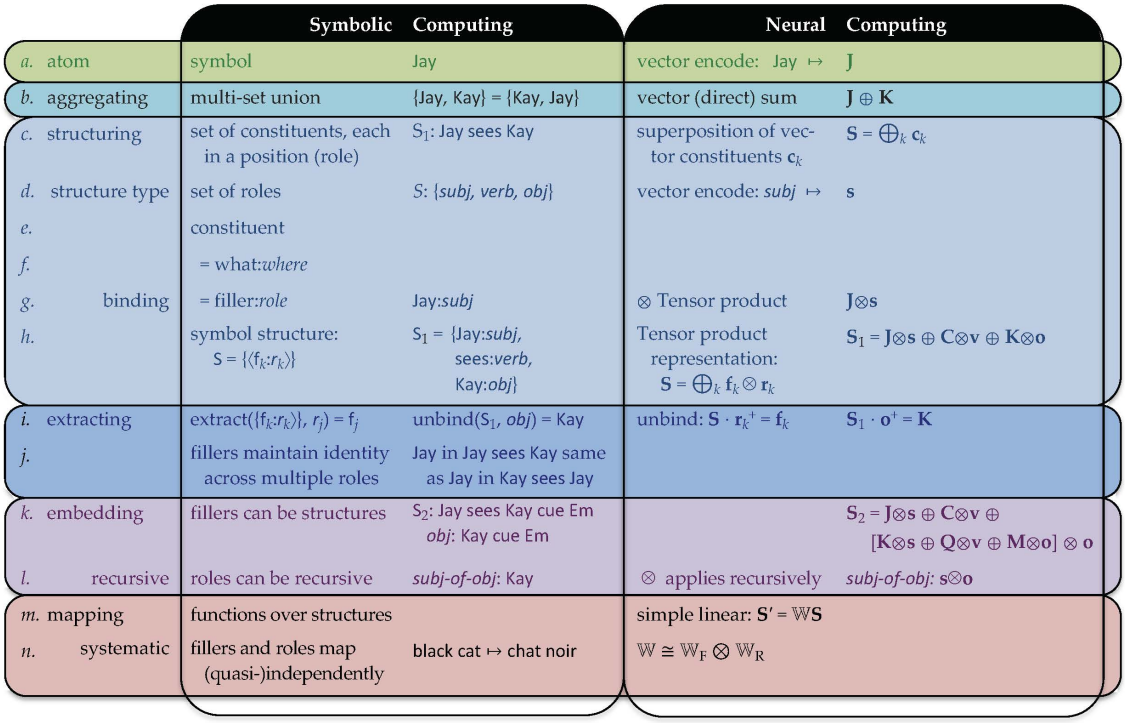

Figure 9: Encoding structural primitives of compositional-structure processing in symbolic and neural computing.

Figure 9: Neurally encoding structural primitives of compositional computing

Left column. Structuring information compositionally starts with atomic elements (***a***), aggregates them into groups (***b***), and arranges them into distinct, inter-related roles (***c***). Thus a compositional structure is a collection of constituents (***h***), each of which has a filler ('what') and a role ('where') aspect (***e–f***) that are bound together (***g***).

A structure for a simple sentence consists in three roles: *subject*, *object*, *verb* (***d***). A particular instance of that structure, Jay sees Kay, is a collection of filler:*role* bindings: Jay:*subj*, Kay:*obj*, sees:*verb*. Given a structure, the filler of a specified role can be extracted (or unbound) (***i***); crucially, the filler retains its identity as it is bound to different roles (***j***). Structures can be embedded within larger structures (***k***): the filler of a role can itself be a structure, like the role *obj* in Jay sees Kay cue Em which is filled by the structure Kay cue Em; Kay's role is *subj-of-obj*—roles can be recursive (***l***).

Computation over structures maps an input structure to an output structure (***m***). Such functions can be (fully or partially) *systematic*—the fillers and roles can be mapped (completely or largely) independently (***n***), as in the mapping from the English Adjective Phrase (AP) black cat to French chat noir: fillers black, cat are mapped to noir, chat while roles *AP-L, AP-R* are mapped to *AP-R, AP-L*, respectively (§2.1.2, p. 29).

Middle column. The middle column (left large oval) shows how these conceptual primitives of compositional-structure processing are formalized in symbolic computing.

Right column. The right column (right large oval) shows how these same primitives are vector-encoded in Neurally-Embedded Compositionally-Structured Tensor (NESCT) computing, one approach to neurocompositional computing. These neurally-encoded structures are Tensor Product Representations (TPRs; ***h***), named after the operation used to bind neurally-encoded fillers (***a***) to neurally-encoded roles (***d***): the tensor product $\otimes$, defined in the inset of Fig. 3, p. 25, (***g***). This is a generalization of the outer product of matrix algebra that allows recursive application (e.g., $[(1,3,2)\otimes(0.1,10)]\otimes(1,-1) = (((0.1, 0.3, 0.2), (10, 30, 20)), ((-0.1, -0.3, -0.2), (-10, -30, -20)))$; the elements of the order-3 tensor $\mathbf{u}\otimes\mathbf{v}\otimes\mathbf{w}$ are all the possible products of the elements of $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$).

Aggregating filler:*role* bindings is done with $\oplus$, the direct sum (***b–c***): a generalization of simple vector addition that allows for the addition of tensors of different order. A filler can itself be an entire structure; this gives rise to multiple tensor products for the multiple levels of embedding in the structure (***k–l***). The general TPR is a concatenation (or list) of tensors of order 1 (encoding symbols at embedding depth 1), order 2 (encoding symbols at embedding depth 2), etc. (***k***), and the direct sum simply applies ordinary vector addition separately within the subspaces of tensors of different orders. (For an alternative approach to TPRs for trees, see [64].)

Extracting fillers from a TPR—*unbinding* a role to identify its filler—is done with a tensor inner product (***i–j***). To unbind any of a set of n role encodings $\{\mathbf{r}_i\}_{i=1}^n$, we need the *unbinding* (or dual) vectors $\{\mathbf{r}_i^+\}_{i=1}^n$ defined by the requirement that $\mathbf{r}_j\cdot\mathbf{r}_k^+ = \delta_{jk} = 1$ if $j=k$ and 0 otherwise. A set of dual vectors exists if the number of neurons that encode the roles is sufficient (at least as large as the number of possible roles n). Then, from the TPR $\mathbf{S}$ of a structure S in which the role encodings $\{\mathbf{r}_i\}_{i=1}^n$ are bound to the particular filler encodings $\{\mathbf{f}_i\}_{i=1}^n$, we can unbind any $\mathbf{r}_k$ by taking the inner product of $\mathbf{S}$ and $\mathbf{r}_k^+$: $[\mathbf{S}\cdot\mathbf{r}_k^+]_i = \sum_{j=1}^n \mathbf{S}_{ij}[\mathbf{r}_k^+]_j = [\mathbf{f}_k]_i$.

Mathematical results show that complex recursive symbolic functions can be computed by networks on entire TPR encodings, operating on all embedded constituents in parallel [206]. Many interesting functions (like the one computed by LFNET, right half of Fig. 2, p. 20) can even be performed by a linear network, which simply multiplies an input TPR $\mathbf{S}$ by a weight matrix $\mathbb{W}$ to produce an output TPR $\mathbf{S}'$ (***m***). Independent transformation of fillers and roles (***n***)—pure systematicity—arises if this weight matrix $\mathbb{W}$ factors as the tensor product of two smaller matrices, one that transforms fillers ($\mathbb{W}_F$) and the other that transforms roles ($\mathbb{W}_R$).

7 Appendix B. COPYNET experiments

In the COPYNET experiments illustrated in Figure 4, p. 37, we trained models on the task of copying sequences of digits: Given a five-digit sequence as input (e.g., ⟨3,9,7,4,7⟩), the model had to produce the same sequence as its output. During training, models encountered all digits from 0 to 9 and all positions from first to fifth, but certain digit/position pairs were withheld (e.g., 3 in the third position). Correctly handling such sequences therefore required compositional generalization. The code for these experiments is available at `https://github.com/tommccoy1/copynet`; below we provide more details about the experiments.

7.1 Data

We divided sequences into two types: n-in-*n* sequences, in which at least one position in the sequence is occupied by the digit corresponding to that position (e.g., ⟨3, 9, 7, **4**, 7⟩ contains 4 in position *4*), and non-n-in-*n* sequences, which contain no n-in-*n* digits. We generated 10,000 non-n-in-*n* training sequences, 1,000 non-n-in-*n* validation sequences, 1,000 non-n-in-*n* test sequences, and 1,000 n-in-*n* test sequences. Because all training sequences were non-n-in-*n* sequences, the n-in-*n* test items required deeper generalization from the training set than the non-n-in-*n* test items did. All sequences were 5 digits long and were drawn randomly from the set of sequences satisfying the relevant n-in-*n* or non-n-in-*n* constraint. No sequences were repeated within or across data splits. The digits in the sequences could have values ranging from 0 to 9 inclusive.

7.2 Models

We trained three types of neural-network models on the copying task: LSTMs, Transformers, and NECSTransformers. All three were structured as sequence-to-sequence models, in which the model has two sub-networks: an *encoder*, which receives a sequence as input and converts it to a vector representation called the encoding, and a *decoder*, which takes in the encoding and uses it to produce the output sequence.

LSTMs [76] are a type of recurrent neural network [51, 90]. An LSTM encoder takes in the input sequence one element at a time and stores information about what it has seen so far in a pair of vectors (the hidden state and cell state), where these vectors are updated after each input element. The hidden state and cell state at the end of the input together serve as the encoding. The LSTM decoder then takes in the encoding and produces output elements one at a time. The LSTMs that we used did not involve attention.

In Transformers [229], the encoder and decoder each feature several layers of processing. In the encoder, there is one vector for each input element. The vectors at a given layer are produced by selectively combining and modifying information from all of the vectors in the previous layer using a mechanism called attention [5]. The collection of vectors in the final encoder layer serves as the encoding. The decoder then produces the output one element at a time; at each time step of decoding, the decoder can access information from the full set of input encoding vectors as well as the vector representations of the previously-produced output tokens.

The NECSTransformer [187] is a Transformer enhanced to incorporate Tensor Product Representations (see §2.2.2.2, p. 36). Where the standard Transformer produces a vector representing each token, the TP-Transformer additionally binds this token representation to a role vector so that the TP-Transformer can explicitly represent each token's role in the sequence. The NECSTransformer is not provided with any information about *which* roles it should use or how to encode them; it must construct its own role scheme through learning.

In the plots in Fig. 4, p. 37, we refer to the LSTM as CopyNet-0 because it is non-neurocompositional, the Transformer as CopyNet-1G because it uses first-generation neurocompositional computing, and the NECSTransformer as CopyNet-2G because it uses second-generation neurocompositional computing.

The LSTMs had a hidden and embedding size of 257; the Transformers had a hidden and embedding size of 256; and the NECSTransformers had a hidden and embedding size of 230. These varying values were chosen so that all models would have similar overall parameter counts: 2,131,056 for LSTMs, 2,115,584 for Transformers, and 2,115,366 for NECSTransformers. All models used 2 hidden layers and had dropout [217] applied between all layers with a dropout probability of 0.1. The Transformers and NECSTransformers both used 4 attention heads and a feedforward dimensionality of 256. When producing outputs, all models generated tokens until they produced a special end-of-sequence token, which signalled that they had finished their output.

7.3 Training

Models were trained using negative log likelihood loss over the correct output tokens, using the Adam optimizer [98] with a learning rate of 0.0001 and a batch size of 10. Teacher forcing was used during training but not evaluation. Models could loop over their training examples as many times as necessary for them to converge. After every complete pass over the model's training examples, each model was evaluated on the validation set. Training halted when, for 5 consecutive validation set evaluations, the loss had not improved. In all evaluations, a model-produced output sequence was only counted as correct if it exactly matched the correct output sequence, digit-for-digit; no credit was given for partially-correct outputs.

7.4 Learning curve experiments

To produce the learning curves shown in Figs. 4*a* and 4*b*, we trained models on subsets of the training set. The subset sizes that we used were 1, multiples of 10 from 10 to 200 inclusive, multiples of 50 from 250 to 1000 inclusive, and multiples of 100 from 1000 to 2000 inclusive. For each subset of size k, the specific examples used were the first k examples in the training set, meaning that each successively larger k yields a strict superset of the examples used in smaller values of k. For each model and training set size, we performed 10 re-runs with different random seeds. The y-axis value for each point in Figs. 4*a* and 4*b* is the mean across all 10 re-runs.

7.5 Full training set experiments

For each model type, we also performed 100 runs of training the model on the full training set (10,000 examples) to generate the plots in Fig. 4*c* and 4*d*. We evaluated these models

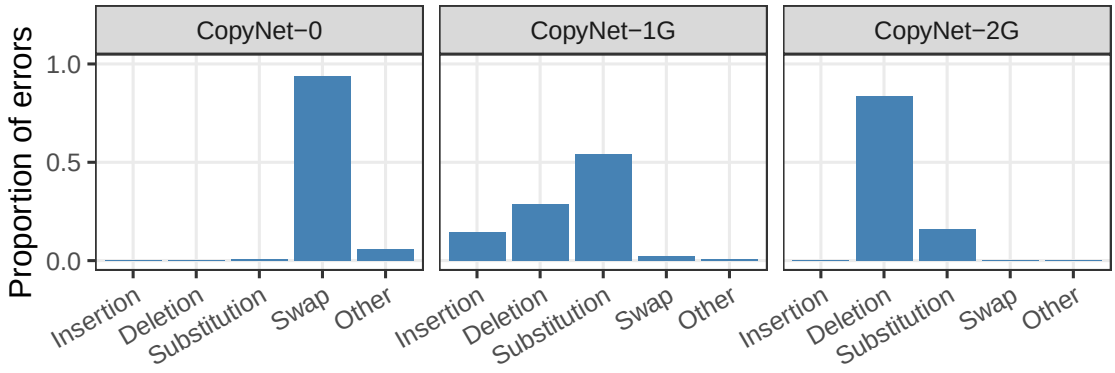

Figure 10: Categorization of errors made by CopyNet models. Within each model category, all bars sum to 1.

not on the 1,000-item test sets but rather on the complete set of all 100,000 possible 5-digit sequences. Fig. 4*c* shows results on this set of 100,000 examples, where each bar is the mean across the 100 runs of the model. Fig. 4*d* shows the proportion of the 100 runs of each model that attained perfect performance—producing the correct output for all 100,000 possible inputs.

7.6 Analysis of errors

In addition to the results shown in Fig. 4, here we also categorize the types of errors that models make. Though errors are rare overall, all models make at least some errors in at least some runs. We consider 5 error categories:

- **Insertion:** Producing the correct output except with a single additional digit inserted (e.g., ⟨3,9,**6**,7,4,7⟩ instead of ⟨3,9,7,4,7⟩).
- **Deletion:** Producing the correct output except with a single digit deleted (e.g., ⟨3,9,7,4⟩ instead of ⟨3,9,7,4,**7**⟩).
- **Substitution:** Producing the correct output except with a single digit replaced by some other digit (e.g., ⟨**0**,9,7,4,7⟩ instead of ⟨**3**,9,7,4,7⟩).
- **Swap:** Producing the correct output except with two digits swapped (e.g., ⟨**4**,9,7,**3**,7⟩ instead of ⟨**3**,9,7,**4**,7⟩).
- **Other:** Errors not falling into any of the above categories, such as inserting more than one digit.

The breakdown by category for each model type's errors is in Fig. 10. CopyNet-0's errors are typically swaps; CopyNet-1's errors are most often substitutions; and CopyNet-2's errors are usually deletions. The differences in these error patterns may arise from differences in the strategies that the models use for encoding information, but we leave the investigation of this possibility for future work.

8 Appendix C. Online sentence processing in a NECST network

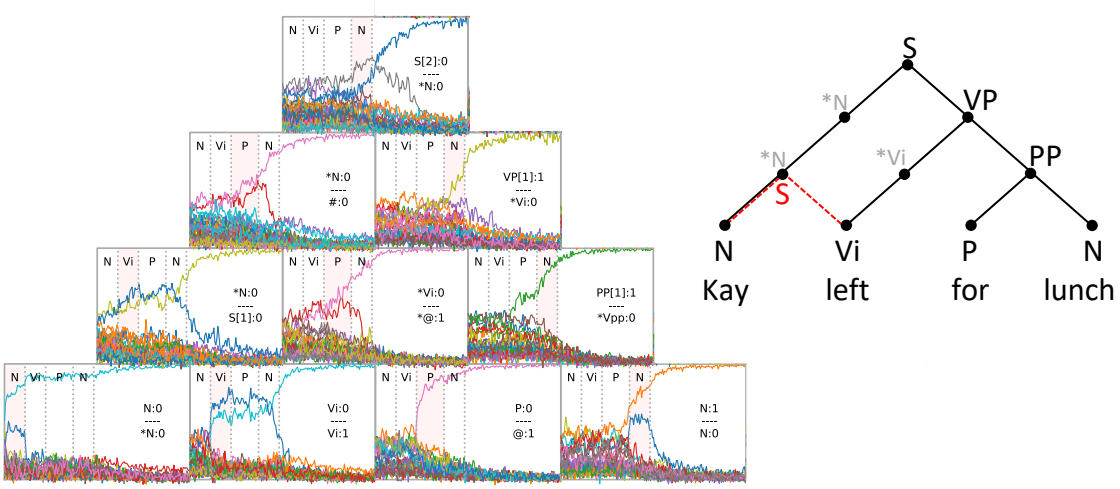

Figure 11: Real-time sentence processing in a NECST network [29]

Figure 11: Real-time sentence processing in a NECST network: ParseNet

The NECST model ParseNet [29, 30] can produce and interpret sentences governed by a grammar encoded in its connections. Here we consider its use in a first stage of sentence comprehension, constructing a parse tree showing the embedded constituent-phrase structure of a string of words presented one at a time. The distributed vector encodings of all symbols in the tree are superimposed upon each other and spread over the entire network, so to 'see' the tree being constructed, the modeler must mathematically analyze the network state at a large scale, as schematically depicted by the telescopes and thought bubbles in Fig. 1, p. 9 and visualized more precisely in the right half of Fig. 3, p. 25.

On the left of Fig. 11 is a plot of a ParseNet's continuous computation over time of (the neural encoding of) the discrete tree structure on the right. In the activation plot, each of the 10 rectangular blocks corresponds to a tree-position role indicated by the dots labeled by symbols on the tree at the right. (We will use the shortcut 'X role' to refer to a position filled by the symbol X in the tree shown.) The modeler analyzes the network state (views it through a mathematical telescope) as it evolves continuously over time, computing the activation of each filler (symbol) in each role (position): the activation of each possible filler:*role* binding is one of the colored curves in the plot. Within each block, the full processing time for the entire sentence is shown on the x-axis; activation is on the y-axis. In this simulation, the network received the input sequence N Vi P N (Noun Verb[intransitive] Preposition Noun), the grammar's representation of a sentence such as Kay left for lunch. One word is presented to the network at a time; within each block, the time interval during which each of the words was presented is demarcated by dashed vertical

lines.

First block in the lowest row (corresponding to the tree role filled by the first N, Kay). While the first word is presented, we see (during the pink-shaded time interval) that one filler rises to dominate the activity: this is the symbol N, the upper symbol displayed at the right of the block (the lower symbol, *N, is the second-most active filler, which rises to medium activity before falling to activity zero; this symbol is used to propagate the N label up the tree to its true parent node, S = sentence). The :0 after N indicates that N is the left child of its parent node; :1 denotes a right child.

Second block in the lowest row (the 'Vi role'). During the second time interval (now marked by the pink band), the input is Vi; the fillers that rise to prominent activity levels are Vi:0 and Vi:1. Given only the first two words, N Vi, the grammar is ambiguous: this is a complete sentence on its own (Kay left), but may be the beginning of a longer sentence. The first possibility is shown by the small red dashed tree on the right; in this case, Vi is a right child (of S): Vi:1. In the second possibility, Vi is a left child (ultimately, of VP = Verb Phrase): Vi:0. As the third and fourth words are presented, the competition resolves in favor of the black tree (Vi:0): the red tree (Vi:1) fades out as the last word is presented.

First block of the second-lowest row. Simultaneously, during presentation of the second word, in the '*N/S role', both possibilities are entertained: the S symbol (from the red tree) has highest activation, but just below is the symbol *N (from the black tree). As later words are presented, this symbol *N overtakes S as required in the grammatically correct parse (black tree) of the full sentence.

Each of the remaining blocks. In each role, the correct filler rises to full activation as the sentence is processed one word at a time (e.g., the PP = Prepositional Phrase filler in the 'PP role'). (Technical details: S[1] is one type of sentence in the grammar—the type of the red tree—while S[2] is another—the type of the black tree; similarly for PP[1] and VP[1]. @ is the 'empty symbol' used for a role with no proper filler; this would correctly apply to the 'P role' if the input actually had only two words.)

Acknowledgments

We gratefully acknowledge, for crucial support and valuable conversations, Johannes Gehrke, Li Deng, Qi Lu, Yi-Min Wang, Harry Shum, Eric Horvitz, Susan Dumais, Xiaodong He, Aslı Çelikyılmaz, Chris Meek, Hamid Palangi, Qiuyuan Huang, Nebojsa Jojic, Imanol Schlag, Kezhen Chen, Shuai Tang, Laurel Brehm, Najoung Kim, Matthias Lalisse, Paul Soulos, Eric Rosen, Caitlin Smith, Coleman Haley, Géraldine Legendre, Jason Eisner, Ben Van Durme, Alan Yuille, Hynek Hermansky, Tal Linzen, Robert Frank, Jürgen Schmidhuber, Ken Forbus, Gary Marcus, Yoshua Bengio, Steven Pinker, Jay McClelland, Alan Prince, Ewan Dunbar, Dapeng Wu, Randy O'Reilly, François Charton, Guillaume Lample, Peter beim Graben, Daniel Crevier, and all our collaborators on the papers reviewed here. The work reported here was supported in part by NSF (GRFP 1746891, BCS-1344269, DGE-0549379) and by Microsoft Research. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or Microsoft.

References

[1] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for Boltzmann Machines. *Cognitive Science*, 9:147–169, 1985.

[2] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE access*, 6:52138–52160, 2018.

[3] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48, 2016.

[4] Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, and al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 2021. 10.1126/science.abj8754.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.

[6] Marco Baroni. Linguistic generalization and compositionality in modern artificial neural networks. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 375(1791):20190307, 2020.

[7] Joost Bastings, Marco Baroni, Jason Weston, Kyunghyun Cho, and Douwe Kiela. Jump to better conclusions: SCAN both left and right. *arXiv preprint arXiv:1809.04640*, 2018.

[8] Peter Beim Graben and Roland Potthast. A dynamic field account to language-related brain potentials. *Principles of Brain Dynamics: Global State Interactions*, 2012.

[9] Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and alternatives. *Computational Linguistics*, 2021. arXiv preprint arXiv:2102.12452.

[10] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623, 2021.

[11] Yoshua Bengio. The consciousness prior. *arXiv preprint arXiv:1709.08568*, 2017.

[12] Yoshua Bengio, Yann Lecun, and Geoffrey Hinton. Deep learning for AI. *Communications of the Association for Computing Machinery*, 64(7):58–65, June 2021.

[13] Rens Bod, Jennifer Hay, and Stefanie Jannedy. *Probabilistic linguistics*. MIT Press, 2003.

[14] Bertram Vivian Bowden. *Faster than thought: A symposium on digital computing machines*. Pitman Publishing, Inc., 1953.

[15] Laurel Brehm, Pyeong Whan Cho, Paul Smolensky, and Matthew A. Goldrick. PIPS: a parallel planning model of sentence production. *Cognitive Science*, in press.

[16] Laurel Brehm and Matthew Goldrick. Distinguishing discrete and gradient category structure in language: Insights from verb-particle constructions. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 43(10):1537, 2017.

[17] Marc Brockschmidt, Miltiadis Allamanis, Alexander L Gaunt, and Oleksandr Polozov. Generative code modeling with graphs. In *International Conference on Learning Representations*, 2019.

[18] Allan G Bromley. Charles Babbage's analytical engine, 1838. *IEEE Annals of the History of Computing*, 20(4):29–45, 1998.

[19] Rodney Brooks. Predictions scorecard, 2021 January 01, 2021. https://rodneybrooks.com/predictions-scorecard-2021-january-01/, retrieved Nov. 15, 2021.

[20] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[21] Joan Bybee and James L. McClelland. Alternatives to the combinatorial paradigm of linguistic theory based on domain general principles of human cognition. *The Linguistic Review*, 22(2–4), 2005.

[22] Paco Calvo and John Symons. *The architecture of cognition: Rethinking Fodor and Pylyshyn's systematicity challenge*. MIT Press, 2014.

[23] Supriyo Chakraborty, Richard Tomsett, Ramya Raghavendra, Daniel Harborne, Moustafa Alzantot, Federico Cerutti, Mani Srivastava, Alun Preece, Simon Julier, Raghuveer M. Rao, Troy D. Kelley, Dave Braines, Murat Sensoy, Christopher J. Willis, and Prudhvi Gurram. Interpretability of deep learning models: A survey of results. In *Proceedings of the 2017 IEEE SmartWorld Conference*, pages 1–6, 2017.

[24] R. Chandrasekar. Elementary? Question answering, IBM's Watson, and the Jeopardy! challenge. *Resonance*, 19:221–241, 2014.

[25] Kezhen Chen, Qiuyuan Huang, Hamid Palangi, Paul Smolensky, Kenneth D. Forbus, and Jianfeng Gao. Mapping natural-language problems to formal-language solutions using structured neural representations. In *International Conference on Machine Learning*, pages 1566–1575. PMLR, 2020.

[26] Jean-Pierre Chevrot, Céline Dugua, and Michel Fayol. Liaison acquisition, word segmentation and construction in french: a usage-based account. *Journal of child language*, 36(3):557–596, 2009.

[27] Kyunghyun Cho, Bart Van Merriënboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014. arXiv preprint arXiv:1406.1078.

[28] Pyeong Whan Cho, Matthew Goldrick, Richard L Lewis, and Paul Smolensky. Dynamic encoding of structural uncertainty in gradient symbols. In *Proceedings of*

the 8th Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2018), pages 19–28, 2018.

[29] Pyeong Whan Cho, Matthew Goldrick, and Paul Smolensky. Incremental parsing in a continuous dynamical system: Sentence processing in gradient symbolic computation. *Linguistics Vanguard*, 3(1), 2017.

[30] Pyeong Whan Cho, Matthew Goldrick, and Paul Smolensky. Parallel parsing in a gradient symbolic computation parser. *PsyArXiv/utcgv*, 2020.

[31] Noam Chomsky. *Aspects of the theory of syntax*. MIT Press, 1965.

[32] Noam Chomsky. *Lectures on government and binding*. Foris, 1981.

[33] Alonzo Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58(2):345–363, 1936.

[34] Patricia Smith Churchland and Terrence Joseph Sejnowski. *The computational brain*. MIT press, 2016.

[35] Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 3882–3890, 2020.

[36] Arthur C Clarke. *2001: A space odyssey*. New American Library, 1968.

[37] Allan Collins and Edward E Smith. *Readings in cognitive science: A perspective from psychology and artificial intelligence*. Morgan Kaufmann, 1988.

[38] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017.

[39] Eric Crawford, Matthew Gingerich, and Chris Eliasmith. Biologically plausible, human-scale knowledge representation. *Cognitive Science*, 40(4):782–821, 2016.

[40] Daniel Crevier. *AI: The tumultuous history of the search for artificial intelligence*. Basic Books, Inc., 1993.

[41] Marie-Hélène Côté. French liaison. In Marc van Oostendorp, Colin J Ewen, Beth Hume, and Keren Rice, editors, *Blackwell Companion to Phonology*, pages 2685–2710. Wiley-Blackwell, Malden, MA, 2011.

[42] Roberta D'Alessandro. The achievements of generative syntax: A time chart and some reflections. *Catalan journal of linguistics*, pages 7–26, 2019.

[43] Marina Danilevsky, Kun Qian, Ranit Aharonov, Yannis Katsis, Ban Kawas, and Prithviraj Sen. A survey of the state of explainable ai for natural language processing. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 447–459, 2020.

[44] Sreerupa Das and Michael C Mozer. A unified gradient-descent/clustering architecture for finite state machine induction. In *Advances in Neural Information Processing Systems*, pages 19–26, 1994.

[45] Ernest Davis. The use of deep learning for symbolic integration: A review of (Lample and Charton, 2019). *arXiv preprint arXiv:1912.05752*, 2019.

[46] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 4171–4186. Association for Computational Linguistics, 2019.

[47] Jacob Devlin, Jonathan Uesato, Surya Bhupatiraju, Rishabh Singh, Abdel-rahman Mohamed, and Pushmeet Kohli. RobustFill: Neural program learning under noisy I/O. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 990–998. JMLR.org, 2017.

[48] Yao Dou, Maxwell Forbes, Rik Koncel-Kedziorski, Noah A Smith, and Yejin Choi. Scarecrow: A framework for scrutinizing machine text. *arXiv preprint arXiv:2107.01294*, 2021.

[49] Stewart Duncan. Thomas Hobbes. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2021 edition, 2021.

[50] Chris Eliasmith. *How to build a brain: A neural architecture for biological cognition*. Oxford University Press, 2013.

[51] Jeffrey L Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[52] Roland Fernandez, Asli Celikyilmaz, Rishabh Singh, and Paul Smolensky. Learning and analyzing vector encoding of symbolic representations. *arXiv preprint arXiv:1803.03834*, 2018.

[53] Jerry A Fodor and Zenon W Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, 1988.

[54] Gottlob Frege and Michael Beaney. *The Frege reader*. Blackwell, 1997.

[55] Daniel Furrer, Marc van Zee, Nathan Scales, and Nathanael Schärli. Compositional generalization in semantic parsing: Pre-training vs. specialized architectures. *arXiv preprint arXiv:2007.08970*, 2020.

[56] Artur d'Avil Garcez, Marco Gori, L Luciano Serafini Lamb, Michael Spranger, and Son N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *Journal of Applied Logics*, 6(4):611–632, 2019.

[57] Ross W Gayler. Vector symbolic architectures answer jackendoff's challenges for cognitive neuroscience. In *Joint International Conference on Cognitive Science*, 2003.

[58] Mario Giulianelli, Jack Harding, Florian Mohnert, Dieuwke Hupkes, and Willem Zuidema. Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information. In *EMNLP BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, 2018.

[59] Matthew Goldrick, Michael Putnam, and Lara Schwarz. Coactivation in bilingual grammars: A computational account of code mixing. *Bilingualism: Language and Cognition*, 19(5):857–876, 2016.

[60] Edward Grefenstette. Towards a formal distributional semantics: Simulating logical calculi with tensors. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 1–10, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics.

[61] Joshua M Griffiths. On the rapid expansion of optimality theory at the end of the twentieth century. *Historiographia Linguistica*, 46(1-2):133–162, 2019.

[62] Barbara J Grosz and Peter Stone. A century-long commitment to assessing artificial intelligence and its impact on society. *Communications of the Association for Computing Machinery*, 61(12):68–73, 2018.

[63] John Hale and Paul Smolensky. Harmonic grammars and harmonic parsers for formal languages. In Paul Smolensky and Géraldine Legendre, editors, *The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar. Vol. 1: Cognitive Architecture*, pages 393–416. MIT press, 2006.

[64] Coleman Haley and Paul Smolensky. Invertible tree embeddings using a cryptographic role embedding scheme. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3671–3683, 2020.

[65] John Haugeland. *Artificial Intelligence: The very idea*. MIT press, 1989.

[66] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. DeBERTa: Decoding-enhanced BERT with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

[67] James Henderson. The unstoppable rise of computational linguistics in deep learning. In *Association for Computational Linguistics*, 2020.

[68] James A. Hendler, Austin Tate, and Mark Drummond. AI planning: Systems and techniques. *AI magazine*, 11(2):61–61, 1990.

[69] Geoffrey Hinton. How to represent part-whole hierarchies in a neural network. *arXiv preprint arXiv:2102.12627*, 2021.

[70] Geoffrey E. Hinton. Implementing semantic networks in parallel hardware. In Geoffrey E. Hinton and James A. Anderson, editors, *Parallel Models of Associative Memory*, pages 201–232. Erlbaum, 1981.

[71] Geoffrey E. Hinton. Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, 1986.

[72] Geoffrey E. Hinton and James A. Anderson, editors. *Parallel models of associative memory*. Erlbaum Publishers, 1981.

[73] Geoffrey E Hinton, James L McClelland, and David E Rumelhart. Distributed representations. In David E. Rumelhart, James L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the microstructure of cognition: Vol. 1, Foundations*, pages 77–109. MIT Press, 1986.

[74] Geoffrey E Hinton and Terrence Sejnowski, editors. *Unsupervised learning: Foundations of neural computation*. MIT press, 1999.

[75] Wolfram Hinzen, Edouard Machery, and Markus Werning, editors. *The Oxford Handbook of Compositionality*. Oxford University Press, 2012.

[76] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[77] Douglas R. Hofstadter. *Gödel, Escher, Bach: An Eternal Golden Braid*. Basic Books, 1979.

[78] Keith J Holyoak and John E Hummel. The proper treatment of symbols in a connectionist architecture. In Eric Dietrich and Arthur B. Markman, editors, *Cognitive dynamics: Conceptual and representational change in humans and machines*, pages 229–264. Psychology Press, 2000.

[79] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[80] Qiuyuan Huang, Li Deng, Dapeng Wu, Chang Liu, and Xiaodong He. Attentive tensor product learning. In *33rd AAAI Conference on Artificial Intelligence*, pages 1344–1351, 2019.

[81] Qiuyuan Huang, Paul Smolensky, Xiaodong He, Li Deng, and Dapeng Wu. Tensor product generation networks for deep NLP modeling. In *North American Association for Computational Linguistics*, 2018.

[82] John E Hummel, Keith J Holyoak, Collin B Green, Leonidas A A Doumas, Derek Devnich, Aniket Kittur, and Donald J Kalar. A solution to the binding problem for compositional connectionism. In Simon D Levy and Ross Gayler, editors, *Compositional Connectionism in Cognitive Science: Papers from the AAAI Fall Symposium*, pages 31–34, 2004.

[83] Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795, 2020.

[84] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[85] Sarthak Jain and Byron C. Wallace. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[86] Theo M. V. Janssen. Compositionality: Its historical context. In Wolfram Hinzen, Edouard Machery, and Markus Werning, editors, *The Oxford Handbook of Compositionality*. Oxford University Press, 2012.

[87] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In *57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[88] Yichen Jiang, Asli Celikyilmaz, Paul Smolensky, Paul Soulos, Sudha Rao, Hamid Palangi, Roland Fernandez, Caitlin Smith, Mohit Bansal, and Jianfeng Gao. Enriching transformers with structured tensor-product representations for abstractive summarization. In *North American Chapter of the Association for Computational Linguistics*, pages 4780–4793, 2021.

[89] Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. Estimators for stochastic "unification-based" grammars. *arXiv preprint cs/0008028*, 2000.

[90] Michael I Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the 1986 Cognitive Science Conference*, pages 531–546. Lawrence Erlbaum, 1986.

[91] Aravind K. Joshi. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In David R. Dowty, Lauri Karttunen, and Arnold M. Zwicky, editors, *Natural language parsing: Psychological, computational, and theoretical perspectives*, pages 206–250. Cambridge University Press, 1985.

[92] Aravind K. Joshi, Leon S. Levy, and Masako Takahashi. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1):136–163, 1975.

[93] René Kager. *Optimality theory*. Cambridge university press, 1999.

[94] Daniel Kahneman. *Thinking, fast and slow*. Macmillan, 2011.

[95] Pentti Kanerva. Binary spatter-coding of ordered k-tuples. In *International Conference on Artificial Neural Networks*, pages 869–873. Springer, 1996.

[96] Najoung Kim and Tal Linzen. COGS: A compositional generalization challenge based on semantic interpretation. In *Empirical Methods in Natural Language Processing*, November 2020.

[97] Najoung Kim, Kyle Rawlins, and Paul Smolensky. The complement-adjunct distinction as gradient blends: The case of English prepositional phrases. *https://ling.auf.net/lingbuzz/004723*, 2019.

[98] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference for Learning Representations*, 2015.

[99] Paul Kiparsky and Johan F Staal. Syntactic and semantic relations in Pāṇini. *Foundations of Language*, pages 83–117, 1969.

[100] Ronald Kline. Cybernetics, automata studies, and the Dartmouth conference on artificial intelligence. *IEEE Annals of the History of Computing*, 33(4):5–16, 2010.

[101] Nitin Lahoti. Machine learning in mobile applications: The next wave of enterprise mobility. *Mobisoft blog*, 2020. /mobisoftinfotech.com/resources/blog/machine-learning-in-mobile-applications/, retrieved Oct. 31, 2021.

[102] Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *35th International Conference on Machine Learning (ICML 2018)*, 2017.

[103] Brenden M Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2879–2888, 2018.

[104] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40:e253, 2017.

[105] Matthias R Lalisse and Paul Smolensky. Augmenting compositional models for knowledge base completion using gradient representations. In *Society for Computation in Linguistics*, pages 257–266, 2019.

[106] Guillaume Lample and François Charton. Deep learning for symbolic mathematics. *arXiv preprint arXiv:1912.01412*, 2019.

[107] Thomas K. Landauer and Susan T. Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211–240, 1997.

[108] Karim Lari and Steve J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language*, 4(1):35–56, 1990.

[109] Yann LeCun. Deep learning hardware: Past, present, and future. In *IEEE International Solid-State Circuits Conference*, pages 12–19, 2019.

[110] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[111] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[112] Géraldine Legendre, Jane Grimshaw, and Sten Vikner. *Optimality-theoretic syntax*. MIT Press, 2001.

[113] Géraldine Legendre, Yoshiro Miyata, and Paul Smolensky. Harmonic grammar—A formal multi-level connectionist theory of linguistic well-formedness: Theoretical foundations. In *Proceedings of the 12th Meeting of the Cognitive Science Society*, pages 388–395, 1990.

[114] Géraldine Legendre, Yoshiro Miyata, and Paul Smolensky. Distributed recursive structure processing. In *Advances in Neural Information Processing Systems*, pages 591–597, 1991.

[115] Géraldine Legendre, Michael T Putnam, Henriette De Swart, and Erin Zaroukian. *Optimality-theoretic syntax, semantics, and pragmatics: From uni- to bidirectional optimization*. Oxford University Press, 2016.

[116] Simon D Levy and Ross Gayler. Vector symbolic architectures: A new building material for artificial general intelligence. In *Proceedings of the Conference on Artificial General Intelligence*, pages 414–418, 2008.

[117] Gideon Lewis-Kraus. The great AI awakening. *The New York Times Magazine*, 14:12, 2016.

[118] Belinda Z. Li, Maxwell Nye, and Jacob Andreas. Implicit representations of meaning in neural language models. *arXiv preprint arXiv:2106.00737*, 2021.

[119] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

[120] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *arXiv preprint arXiv:2106.04554*, 2021.

[121] Xi Victoria Lin, Chenglong Wang, Deric Pang, Kevin Vu, Luke Zettlemoyer, and Michael D Ernst. Program synthesis from natural language using recurrent neural networks. *University of Washington Department of Computer Science and Engineering, Seattle, WA, USA, Tech. Rep. UW-CSE-17-03-01*, 2017.

[122] Tal Linzen. Issues in evaluating semantic spaces using word analogies. *arXiv preprint arXiv:1606.07736*, 2016.

[123] Tal Linzen. How can we accelerate progress towards human-like linguistic generalization? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5210–5217, 2020.

[124] Tal Linzen, Emmanuel Dupoux, and Benjamin Spector. Quantificational features in distributional word representations. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 1–11, 2016.

[125] Cynthia MacDonald and Graham MacDonald. *Debates on psychological explanation: Connectionism*. Blackwell Publishers, 1995.

[126] Christopher D. Manning. Stanford parser, 2017. `nlp.stanford.edu/software/lex-parser.shtml`.

[127] Christopher D. Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117(48):30046–30054, 2020.

[128] Gary Marcus. *The algebraic mind: Integrating connectionism and cognitive science*. MIT press, 2001.

[129] Gary Marcus. Innateness, AlphaZero, and artificial intelligence. *arXiv preprint arXiv:1801.05667*, 2018.

[130] Gary Marcus. The next decade in AI: Four steps towards robust artificial intelligence. *arXiv preprint arXiv:2002.06177*, 2020.

[131] Gary Marcus and Ernest Davis. *Rebooting AI: Building artificial intelligence we can trust*. Pantheon, 2019.

[132] James L McClelland and Joan Bybee. Gradience of gradience: A reply to jackendoff. *The Linguistic Review*, 24(4), 2007.

[133] R. Thomas McCoy, Erin Grant, Paul Smolensky, Thomas L. Griffiths, and Tal Linzen. Universal linguistic inductive biases via meta-learning. In *Proceedings of the 42nd Annual Meeting of the Cognitive Science Society*, 2020.

[134] R Thomas McCoy, Tal Linzen, Ewan Dunbar, and Paul Smolensky. RNNs implicitly implement tensor product representations. In *International Conference on Learning Representations*, 2019.

[135] R. Thomas McCoy, Junghyun Min, and Tal Linzen. BERTs of a feather do not generalize together: Large variability in generalization across models with similar test set performance. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 217–227, 2020.

[136] R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, 2019.

[137] Brian P. McLaughlin. Can an ICS architecture meet the systematicity and productivity challenges? In Paco Calvo and John Symons, editors, *The architecture of cognition: Rethinking Fodor and Pylyshyn's systematicity challenge*, pages 31–76. MIT Press Cambridge, MA, 2014.

[138] Nazarré Merchant and Alan Prince. The mother of all tableaux: Order, equivalence, and geometry in the large-scale structure of Optimality Theory. *Rutgers Optimality Archive 1382*, 2021.

[139] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[140] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[141] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *North American Chapter of the Association for Computational Linguistics: Human language Technologies*, pages 746–751, 2013.

[142] Junghyun Min, R Thomas McCoy, Dipanjan Das, Emily Pitler, and Tal Linzen. Syntactic data augmentation increases robustness to inference heuristics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2339–2352, 2020.

[143] Marvin Minsky. Logical vs. analogical or symbolic vs. connectionist or neat vs. scruffy. *AI Magazine*, 12(2):34–51, 1991.

[144] Marvin L. Minsky. *Computation: Finite and infinite machines*. Prentice-Hall, 1967.

[145] Jeff Mitchell and Mirella Lapata. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429, 2010.

[146] Sarthak Mittal, Sharath Chandra Raparthy, Irina Rish, Yoshua Bengio, and Guillaume Lajoie. Compositional attention: Disentangling search and retrieval. *arXiv preprint arXiv:2110.09419*, 2021.

[147] Richard Montague. Universal grammar. *Theoria*, 36(3):373–398, 1970.

[148] Richard Montague. The proper treatment of quantification in ordinary english. In K. J. J. Hintikka, J. M. E. Moravcsik, and P. Suppes, editors, *Approaches to natural language*, pages 221–242. Springer, 1973.

[149] Hans Moravec. *Mind children: The future of robot and human intelligence*. Harvard University Press, 1988.

[150] Michael C. Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *Advances in Neural Information Processing Systems*, pages 107–115, 1989.

[151] Bennet B Murdock. A theory for the storage and retrieval of item and associative information. *Psychological Review*, 89(6):609, 1982.

[152] Max Nelson. Joint learning of constraint weights and gradient inputs in Gradient Symbolic Computation with constrained optimization. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 224–232. Association for Computational Linguistics, 2020.

[153] Allen Newell. Physical symbol systems. *Cognitive Science*, 4(2):135–183, 1980.

[154] Allen Newell, J Clifford Shaw, and Herbert A Simon. The processes of creative thinking. *RAND Corporation Reports*, 1958.

[155] Peter Norvig. On the (small) number of atoms in the universe, 2016. http://norvig.com/atoms, retrieved Nov. 4, 2021.

[156] D. W. Otter, J. R. Medina, and J. K. Kalita. A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21, 2020.

[157] Hamid Palangi, Paul Smolensky, Xiaodong He, and Li Deng. Question-answering with grammatically-interpretable representations. In *American Association for Artificial Intelligence*, pages 5350–5357, 2018.

[158] Joe Pater. Weighted constraints in generative linguistics. *Cognitive Science*, 33(6):999–1035, 2009.

[159] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

[160] Ellie Pavlick. Semantic structure in deep learning. *Annual Review of Linguistics*, 8(1):447–471, 2022.

[161] Francis Jeffry Pelletier. The principle of semantic compositionality. *Topoi*, 13(1):11–24, 1994.

[162] Gerald Penn and Paul Kiparsky. On Pāṇini and the generative capacity of contextualized replacement systems. In *Proceedings of COLING 2012: Posters*, pages 943–950, 2012.

[163] Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[164] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237, 2018.

[165] Joshua C. Peterson, Dawn Chen, and Thomas L. Griffiths. Parallelograms revisited: Exploring the limitations of vector space models for simple analogies. *Cognition*, 205:104440, 2020.

[166] Ray Pike. Comparison of convolution and matrix distributed memory systems for associative recall and recognition. *Psychological Review*, 91(3):281, 1984.

[167] Tony A Plate. Holographic Reduced Representations. *IEEE Transactions on Neural networks*, 6(3):623–641, 1995.

[168] Jordan B Pollack. Implications of recursive distributed representations. In *Advances in Neural Information Processing Systems*, pages 527–536, 1989.

[169] Alan S. Prince and Paul Smolensky. *Optimality Theory: Constraint interaction in generative grammar*. Blackwell Publishers, 1993/2004.

[170] Alan S. Prince and Paul Smolensky. Optimality: From neural networks to universal grammar. *Science*, 275(5306):1604–1610, 1997.

[171] James Pustejovsky and Bran Boguraev, editors. *Lexical semantics: The problem of polysemy*. Oxford University Press, 1997.

[172] Zenon Walter Pylyshyn. *Computation and cognition*. MIT Press, 1984.

[173] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, pages 1–26, 2020.

[174] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[175] Charan Ranganath and Maureen Ritchey. Two cortical systems for memory-guided behaviour. *Nature Reviews Neuroscience*, 13(10):713–726, 2012.

[176] Abhilasha Ravichander, Yonatan Belinkov, and Eduard Hovy. Probing the probing paradigm: Does probing accuracy entail task relevance? In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3363–3377, Online, April 2021. Association for Computational Linguistics.

[177] Frank E Ritter, Farnaz Tehranchi, and Jacob D Oury. ACT-R: A cognitive architecture for modeling cognition. *Wiley Interdisciplinary Reviews: Cognitive Science*, 10(3):e1488, 2019.

[178] Raúl Rojas. Konrad Zuse's legacy: The architecture of the Z1 and Z3. *IEEE Annals of the History of Computing*, 19(2):5–16, 1997.

[179] Eric R. Rosen. Learning complex inflectional paradigms through blended gradient inputs. In *Society for Computation in Linguistics*, pages 102–112, 2019.

[180] Kevin Rowe. How search engines use machine learning: 9 things we know for sure. *Search Engine Journal*, 2021. searchenginejournal.com/ml-things-we-know/, retrieved Oct. 31, 2021.

[181] David E Rumelhart and Adele A Abrahamson. A model for analogical reasoning. *Cognitive Psychology*, 5(1):1–28, 1973.

[182] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. In David E Rumelhart, James L McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the microstructure of cognition: Vol. 1, Foundations*, pages 318–362. MIT Press, 1986.

[183] David E. Rumelhart, James L. McClelland, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the microstructure of cognition: Vol. 1, Foundations*. MIT Press, 1986.

[184] Jacob Russin, Roland Fernandez, Hamid Palangi, Eric Rosen, Nebojsa Jojic, Paul Smolensky, and Jianfeng Gao. Compositional processing emerges in neural networks solving math problems. In *43rd Annual Meeting of the Cognitive Science Society*, pages 1767–1773, 2021. arXiv preprint arXiv:2105.08961.

[185] Jake Russin, Jason Jo, Randall C O'Reilly, and Yoshua Bengio. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *arXiv preprint arXiv:1904.09708*, 2019.

[186] Imanol Schlag and Jürgen Schmidhuber. Learning to reason with third order tensor products. In *Advances in Neural Information Processing Systems*, pages 9981–9993, 2018.

[187] Imanol Schlag, Paul Smolensky, Roland Fernandez, Nebojsa Jojic, Jürgen Schmidhuber, and Jianfeng Gao. Enhancing the transformer with explicit relational encoding for math problem solving. In *NeurIPS Workshop on Context and Composition in Biological and Artificial Neural Systems*, 2019. arXiv:1910.06611v2.

[188] Imanol Schlag, Paul Smolensky, Roland Fernandez, Nebojsa Jojic, Jürgen Schmidhuber, and Jianfeng Gao. Enhancing the transformer with explicit relational encoding for math problem solving. *arXiv preprint arXiv:1910.06611v2*, 2020.

[189] Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander WR Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

[190] David Servan-Schreiber, Axel Cleeremans, and James L McClelland. Graded state machines: The representation of temporal contingencies in simple recurrent networks. *Machine Learning*, 7(2-3):161–193, 1991.

[191] Lokendra Shastri and Venkat Ajjanagadde. From simple associations to systematic reasoning: A connectionist representation of rules, variables and dynamic bindings using temporal synchrony. *Behavioral and Brain Sciences*, 16:417–494, 1993.

[192] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

[193] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[194] Herbert Alexander Simon. *The shape of automation for men and management*, volume 13. Harper & Row New York, 1965.

[195] Robin Smith. Aristotle's logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2020 edition, 2020. `plato.stanford.edu/archives/sum2020/entries/aristotle-logic/`.

[196] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In David E Rumelhart, James L McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the microstructure of cognition: Vol. 1, Foundations*, pages 194–281. MIT Press, 1986.

[197] Paul Smolensky. On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 11(1):1–23, 1988.

[198] Paul Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1-2):159–216, 1990.

[199] Paul Smolensky. Harmonic grammars for formal languages. In *Advances in neural information processing systems*, pages 847–854. Citeseer, 1993.

[200] Paul Smolensky. Constituent structure and explanation in an integrated connectionist/symbolic cognitive architecture. In Cynthia MacDonald and Graham MacDonald, editors, *Connectionism: Debates on Psychological Explanation*. Blackwell, 1995.

[201] Paul Smolensky. Computational levels and integrated connectionist/symbolic explanation. In Paul Smolensky and Géraldine Legendre, editors, *The harmonic mind: From neural computation to Optimality-Theoretic grammar. Vol. 2: Linguistic and Philosophical Implications*, pages 503–592. MIT Press, 2006.

[202] Paul Smolensky. Harmony in linguistic cognition. *Cognitive science*, 30(5):779–801, 2006.

[203] Paul Smolensky. Tensor Product Representations: Formal foundations. In Paul Smolensky and Géraldine Legendre, editors, *The harmonic mind: From neural computation to Optimality-Theoretic grammar*, pages 271–344. MIT press, 2006.

[204] Paul Smolensky. Cognition: Discrete or continuous computation? In S. Barry Cooper and Jan van Leeuwen, editors, *Alan Turing – His Work and Impact*, pages 35–41. Elsevier, 2012.

[205] Paul Smolensky. Subsymbolic computation theory for the human intuitive processor. In *Computability in Europe*, pages 675–685. Springer, 2012.

[206] Paul Smolensky. Symbolic functions from neural computation. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 370(1971):3543–3569, 2012.

[207] Paul Smolensky and Matthew Goldrick. Gradient symbolic representations in grammar: The case of French liaison. *Rutgers Optimality Archive*, 1552, 2016.

[208] Paul Smolensky, Matthew Goldrick, and Donald Mathis. Optimization and quantization in gradient symbol systems: A framework for integrating the continuous and the discrete in cognition. *Cognitive Science*, 38(6):1102–1138, 2014.

[209] Paul Smolensky, Moontae Lee, Xiaodong He, Wen-tau Yih, Jianfeng Gao, and Li Deng. Basic reasoning with tensor product representations. *arXiv preprint arXiv:1601.02745*, 2016.

[210] Paul Smolensky and Géraldine Legendre. *The harmonic mind: From neural computation to Optimality-Theoretic grammar. Vol. 1: Cognitive architecture; vol. 2: Linguistic and philosophical implications*. MIT press, 2006.

[211] Paul Smolensky, Géraldine Legendre, and Yoshiro Miyata. Principles for an integrated connectionist/symbolic theory of higher cognition. *University of Colorado Computer Science Technical Reports, CU-CS-600-92*, 1992.

[212] Paul Smolensky, R. Thomas McCoy, Roland Fernandez, Matthew Goldrick, and Jianfeng Gao. Neurocompositional computing: From the Central Paradox of Cognition to a new generation of AI systems. *AI Magazine*, in press. arXiv preprint arXiv:2205.01128.

[213] Paul Smolensky, Eric Rosen, and Matthew Goldrick. Learning a gradient grammar of French liaison. In *Annual Meeting on Phonology*, volume 8, 2020.

[214] Paul Smolensky and Bruce B Tesar. Symbolic computation with activation patterns. In Paul Smolensky and Géraldine Legendre, editors, *The harmonic mind: From neural computation to Optimality-Theoretic grammar. Vol. 1: Cognitive architecture*, volume 1, pages 235–270. MIT press, 2006.

[215] Grace Solomonoff. Ray Solomonoff and the Dartmouth Summer Research Project in Artificial Intelligence, 1956, no date. `raysolomonoff.com/dartmouth/dartray.pdf`.

[216] Paul Soulos, R. Thomas McCoy, Tal Linzen, and Paul Smolensky. Discovering the compositional structure of vector representations with role learning networks. In *Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 238–254, 2020.

[217] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[218] Keith E. Stanovich and Richard F. West. Individual differences in reasoning: Implications for the rationality debate? *Behavioral and Brain Sciences*, 23(5):645–665, 2000.

[219] Terrence Stewart and Chris Eliasmith. Compositionality and biologically plausible models. In Wolfram Hinzen, Edouard Machery, and Markus Werning, editors, *The Oxford Handbook of Compositionality*. Oxford University Press, 2012.

[220] Peter Stone, Rodney Brooks, Erik Brynjolfsson, Ryan Calo, Oren Etzioni, Greg Hager, Julia Hirschberg, Shivaram Kalyanakrishnan, Ece Kamar, Sarit Kraus, et al. Artificial intelligence and life in 2030: The one hundred year study on artificial intelligence, 2016. Stanford University.

[221] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[222] Zoltan Szabó. The case for compositionality. In Wolfram Hinzen, Edouard Machery, and Markus Werning, editors, *The Oxford Handbook of Compositionality*, pages 64–80. Oxford University Press, 2012.

[223] Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, 2019.

[224] Rob Thomas. How AI is driving the new industrial revolution. *Forbes*, 2020.

[225] Michael Tomasello and Raquel Olguin. Twenty-three-month-old children have a grammatical category of noun. *Cognitive development*, 8(4):451–464, 1993.

[226] Paul Tupper, Paul Smolensky, and Pyeong Whan Cho. Discrete symbolic optimization and Boltzmann sampling by continuous neural dynamics: Gradient Symbolic Computation. *arXiv preprint arXiv:1801.03562*, 2018.

[227] Matt Turek. Explainable artificial intelligence (xai). *DARPA website*, 2016. `www.darpa.mil/program/explainable-artificial-intelligence`.

[228] Alan M Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.

[229] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[230] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online, July 2020. Association for Computational Linguistics.

[231] Markus Werning. Non-symbolic compositional representation and its neuronal foundation: Towards an emulative semantics. In Wolfram Hinzen, Edouard Machery, and Markus Werning, editors, *The Oxford Handbook of Compositionality*. Oxford University Press, 2012.

[232] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

[233] James CR Whittington, Timothy H Muller, Shirley Mark, Guifen Chen, Caswell Barry, Neil Burgess, and Timothy EJ Behrens. The Tolman-Eichenbaum machine: Unifying space and relational memory through generalization in the hippocampal formation. *Cell*, 183(5):1249–1263, 2020.

[234] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, 2018.

[235] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[236] Robert Andrew Wilson and Frank C Keil. *The MIT Encyclopedia of the Cognitive Sciences*. MIT press, 2001.

[237] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[238] Ronald Yurko, Francesca Matano, Lee F Richardson, Nicholas Granered, Taylor Pospisil, Konstantinos Pelechrinis, and Samuel L Ventura. Going deep: models for continuous-time within-play valuation of game outcomes in american football with tracking data. *Journal of Quantitative Analysis in Sports*, 16(2):163–182, 2020.

[239] Eva Zimmermann. Gradient symbolic representations and the typology of ghost segments. In *Annual Meeting on Phonology*, volume 7, 2019.