Generative grammar, neural networks, and the implementational mapping problem: Response to Pater

Ewan Dunbar

Generative grammar, neural networks, and the implementational mapping problem: Response to Pater

Ewan Dunbar
CNRS, Université Paris Diderot, Cognitive Machine Learning (Inria) and Université Sorbonne Paris Cité

On the same list that ranked *Syntactic Structures* as the number one most influential twentieth-century work in cognitive science, the number two work cited is David Marr's *Vision* (Marr, 1982). Marr proposed that the only way to achieve a complete understanding of complex information processing systems like those found in the brain is to simultaneously analyze the system at three different levels of analysis: a **computational** level, where one formally defines the problem that the system is solving, specifying which inputs must map to which outputs; as well as an **algorithmic–representational** level, where one spells out a *method* for arriving at an output, given a particular input—a formal hypothesis about how the system encodes information, and how it manipulates it; spelled out at an **implementational** level, which details in what sense the physical system in question can actually be seen as carrying out the proposed algorithm. Generative grammar standardly proposes to analyse language cognition at the first two levels, leaving the problem of how such a system would actually be implemented in the brain open—leaving open, thus, the nature of the link between the algorithmic–representational theory and the physical implementation. Specifying and evaluting any such link is a difficult problem in itself—call it the **implementational mapping problem.**

Pater's proposal for closer interdisciplinary integration between generative grammar and connectionist research will immediately run up against this problem too. Neural network models are not brains, or even brain models, but they are complex systems whose behaviour cannot by understood without some amount of analysis. There is an increasing recognition that we lack good methods for understanding *how* a given neural network model works—for giving a high-level characterization of how the system encodes and manipulates information—and a recognition that, for several reasons, this is an untenable state of affairs. For the time being, this opacity poses serious limits on what we might hope to achieve in a closer integration between connectionist and formal linguistic theories. While we might be able to tell whether and how well a network model is able to learn some linguistic phenomenon, we currently are very limited in our tools for understanding what a given network has learned.

Feedforward neural networks can in principle approximate any function with arbitrary precision. Thus, in principle, they can implement any grammar we can imagine. Clearly, this has the potential to make neural networks extremely useful tools for modelling learning in generative grammar. But their opacity prevents us from arriving at satisfactory answers to basic questions. Has the network arrived at a solution which is (a notational variant of) a grammar made possible under a specific linguistic theory T? Has it arrived at an approximation to such a grammar? Has it arrived at some alternate, unforeseen solution, possible under T, or does its solution fall outside the bounds of T? Can all possible parameter settings learnable in a given network be seen as licit grammars in T, or some approximations to such a grammar? Conversely, can all licit grammars in T be represented or approximated with a given network architecture? In order for the network to learn solutions in the scope of T, does it require special evidence, inaccessible to the child? Or are such grammars always learnable from realistic data? Before we can use neural networks to help us answer these fundamental questions about learnability of grammars, we need to seriously address the problem of how to construct and evaluate a mapping between a grammar defined in a linguistic formalism, on the one hand, and a neural network model, on the other, which may (or may not) encode linguistic knowledge in exactly the way predicted by that grammar.

The implementational mapping problem

Before understanding how the implementational mapping problem is relevant to neural networks, it is useful to look in more detail at Marr's methodological framework, to understand the nature of the problem. Consider the obvious fact that, when a human brain is cut open, trees do not come fanning out. Does this mean that syntax is wrong? No. Even if we had in hand detailed physical state of every neuron in the brain, we would need to do work to understand what they were doing, in the same way that we do now—by proposing and comparing theories. But to do so we would need to have a precise notion what it would mean for neurons to "do something" in a particular way. Assuming that some model of a system correctly describes its inputs and outputs (**what** the system does), how can we determine whether the description at Marr's algorithmic–representational level (**how** it does it) is in really line with the physical object?

Marr's levels of analysis are easily understood using his example of addition of natural numbers. Addition is a function that applies to two natural numbers, and results in some other natural number. There are various kinds of physical systems of which it can be correctly said that they "do addition." A **computational level description** of such a system would be any complete, accurate description of the *behaviour* of the two-place function $add(\cdot, \cdot)$—what kinds of inputs does it take, and what is the relation between its inputs and its outputs? In other words, *what* does it do? There

are various sets of mathematical axioms for arithmetic, some of which are logically equivalent to one another, and others of which define interesting weaker or stronger variants of familiar operations. What they all have in common is that they are sets of formal constraints: the necessary and sufficient conditions for a function $add(\cdot, \cdot)$ to be reasonably called "addition," along with precisions about what rules its inputs and outputs need to follow (in order to be thought of as "numbers"). For example, one such set of axioms, due to Giuseppe Peano, can be given as follows (Mendelson, 1997):

Axioms about numbers

1. The set of numbers (**N**) is a set that contains an element called **0** (among other things)
2. There is a unary function called $successor(\cdot)$ such that
   a. For all $x \in \mathbf{N}$, $successor(x) \in \mathbf{N}$
   b. There is no element $x$of **N** such that $successor(x) = \mathbf{0}$
   c. If $successor(x) = successor(y)$ then $x = y,$ and conversely
3. If some predicate p is true of **0**, and, whenever
   $p(x)$ is true, then $p(successor(x))$ is also true, then
   $p$is true for all elements of **N**

Axioms properly about addition

4. There is a binary function called $add(x, y)$ such that
   a.For all $x \in \mathbf{N}$, $add(x, \mathbf{0}) = x$
   b.For all $x, y \in \mathbf{N}$, $add(x, successor(y)) = successor(add(x, y))$

From these axioms it is easy to prove useful things like, *For all* x, $add(x, successor(\mathbf{0})) = successor(x)$, and that, *For all* x, $add(0, x) = x$, and, from these, that $add(x, y) = add(y, x)$—and all other familiar facts about addition. The axioms precisely state what constraints the inputs must respect, and specify exactly what the output of any addition must be, in relation to its inputs.

A computational-level theory of addition is quite different from a **physical implementation** of addition. This might take the form of a person using an abacus, or doing arithmetic with a pencil and paper, or a spreadsheet taking inputs with a keyboard and giving outputs on a computer monitor. Each of these physical systems include many particles, human brains and fingers, and possibly transistors, beads, or graphite—all of which are rather complicated systems in themselves, all told—but, provided they do nothing other than addition, and do so correctly, then the computational-level theory of addition is an accurate theory of all of their behaviour. The fact that a physical system is doing addition might be intuitively obvious post hoc, *after* the implementational mapping problem has been solved—if you know what addition is, and you have been to school in the West, then you can probably quickly tell when a sequence of scribbles on paper "is" an addition computation. But this background is useful exactly because it tells you *how* to map numbers onto marks on paper, and allows you to recognize the individual steps of a known algorithm for adding them. Without this background, the fact that the computational theory of addition is a correct model of this physical system is not actually "obvious" at all. To convince yourself of this, add some numbers on paper and ask a sample of four-year-olds to explain to you what you are doing. Four-year-olds' inability to give any useful explanation is partly due to their lack of understanding of addition, and partly to a general difficulty in providing useful explanations of things, but also in part due to any knowledge of *what addition could look like.* It does not take much more effort to imagine systems implementing arithmetic which would be completely opaque even to an adult with a solid knowledge of how addition works.[1] For example, consider a person verbally describing the steps of adding two numbers in a foreign language; or an undeciphered clay tablet showing calculations (perhaps using letters of the alphabet as numbers); or a machine that takes two sounds as input, and, if both are pure tones, outputs a new pure tone with a frequency equal to the sum of the two. Any of these, encountered without any knowledge of what they "really" were, would, at first, be inscrutable.

Marr presents a compelling case that to fully understand a system is not only to find a consistent computational-level theory of it, but also to arrive at an understandable, low-level description of the basic physical elements of the implementation—*and* to be able to link the one to the other. Critical to making this link is Marr's second level of description, **algorithmic–representational level.** Going beyond describing *what* a system does, an algorithmic–representational description describes *how* it does it—*without* going into any detail that's tied to a specific physical im-

[1] By this I don't mean knowledge of a formal set of axioms for addition, but simply an everyday understanding of arithmetic. Knowing a set of explicit rules and regulations for arithmetic would be essential to giving a reproducible and externally verifiable scientific demonstration, but an intuitive understanding addition would presumably be sufficient, in most cases, to recognize when a system was doing arithmetic, in the same way that being able to cite all the rules of baseball in their official format is generally not necessary simply in order to be able recognize when a baseball game is being played or described.

plementation. Coming back to addition: how are numbers represented? Several practical possibilities might be decimal numbers ($successor(6) = 7$), binary numbers ($successor(110) = 111$), or unary numbers ($successor(||||||) = |||||||$), but these are not the only possibilities (for example, $successor(\text{VI}) = \text{VII}$). Any one of these systems could be implemented in a variety of physical systems. Unary numbers are familiar because they are so easy to implement: with fingers, sticks, or coins. Roman numerals are normally only encountered in written form—but Roman numerals can be written in upper-case or lower-case, transmitted in Morse code, written using a symbol font—and, of course, stored in a computer as sequences of binary digits. All of these are different physical implementations.

Of course, to properly assess whether a representational system "is" or "is not" a correct theory of a physical system, one must, again, spell out that representational system in some formal detail. To describe a unary encoding, for example, one might simply say that there is a basic element which we write |; that each number in **N** is encoded as a unique sequence consisting only of some number of |'s; and that longer strings always represent larger numbers and conversely. To spell all this out, we would in turn rely on explicit definitions of "sequence" and of "longer," both of which could be mapped to the physical system in some coherent way. A system implementing a unary encoding would not need to contain any actual physical lines written |, but it would need to be coherent with these axioms. A system implementing unary numbers would be different decimal numbers: in the first, it would be true that the physical equivalent of "longer" representations were necessarily larger numbers—but not in the second.

Note that the theory of natural numbers formalized above in axioms (1–3) makes reference to a function called $successor(\cdot)$. This function exists as part of a mathematical definition, but it does not need to have any physical reality: there does not need to be a basic operation "add one" in order for a system to be correctly described by these axioms. A system implementing addition does not even need to have a single physical operation corresponding to the function $add(\cdot, \cdot)$. However, it does need to be capable of doing more than just representing numbers: it needs to be doing addition in some way—using some algorithm. There are many algorithms for doing addition, and the choice of algorithm and the choice of representation are not independent. The familiar column-wise addition with carrying that most children in the West learn for doing addition on paper will work with decimal, binary, or even unary representations; it will not work with Roman numerals. And, although it will work with unary representations, it is not the algorithm one typically uses. The usual method is simple concatenation: $|||| + ||| = |||||||$. But, much to the frustration of young children everywhere, concatenation does not work as a method for adding numbers in decimal representation: $4 + 3$ does not equal 43. An algorithm is a set of implementation-neutral steps for manipulating representations which yields results consistent with the computational-level description of the system. For the algorithmic–representational level description of a system to be correct, there needs to be some consistent mapping between, on the one hand, the physical elements of the system and the changes they undergo in time, and, on the other hand, the basic representational elements and computational steps given by some formally explicit description of an algorithm and a representational system.

The problem of proving that an algorithm does what it is supposed to do at the computational level is a problem familiar to computer science students. The **implementational mapping problem,** on the other hand, is the problem of defining, and empirically evaluating, some mapping between an algorithmic–representational level description, and a physical system. In other words, it is the problem of *knowing what to look for.* This problem is hard. It is made even harder by the fact that Marr's picture is idealized. Systems do not really have only three useful levels of description. A modern computer, for example, is best understood using a large hierarchy of descriptions. I can write Eratosthenes' algorithm for finding prime numbers on a computer. Before being executed, the algorithm will be translated—using another computer program, the compiler—into a sequence of steps rather different from the ones I specified. The compiler will both add steps (when translating my instructions into sequences of instructions understandable by the processor) and remove steps (when it does predictable optimizations). Similarly, I can write an algorithm that simulates doing addition in the way that I learned in school: by representing numbers as strings made up of the characters $0, \ldots, 9$, identifying the "columns," lining them up, adding by column, and carrying when necessary. But each of these strings representing numbers, and each of its component characters, will be stored in the computer's physical memory as binary numbers, using a specific mapping between characters and numbers; translating to and from this representational format represents a system in and of itself. Thus, while it might be correct to say that my program represented numbers in a decimal format, this is only correct if I see my computer memory, as seen from the point of view of my programming language—a set of "objects" corresponding to strings and characters—as the basic "physical" elements of the system—which of course, they are not really. For cognitive science, this suggests that drilling all the way down to the physical implementation level—the human brain—may ultimately require having various intermediate theories of "middle-level" representation. However, even if things were simple in the ideal, the implementational mapping problem would nevertheless be extremely difficult, deserving of careful attention.

The implementational mapping problem in generative grammar

Although my focus here is not on the brain, generative linguists will generally have thought, to one degree or another, about the implementational mapping problem, because of questions about how their theories might be implemented in the brain—the standard "Chomskyan" approach to treat grammars as abstract (not implementation-level) theories of real human language cognition invites these kinds of reflections.

As a point of clarification, however, it is worth pointing out that Marr's division of methodological labour into three levels of analysis is one that does not neatly line up with the way generative grammars are usually given a cognitive interpretation. The standard education in linguistics always includes some insistence on the idea that the series of steps carried out *on paper* to derive a sentence, or a phonological surface form, are not meant as a theory of the series of steps carried out *in the mind.* In other words, no theory of the *algorithm* is implied by the theory of the grammar. Yet, typically, the student is told, the theory of the *representation* (the trees, feature bundles, and so on) *is* supposed to be cognitively intepreted; the derivational steps are merely to be intepreted as a means of stating which representations are licit structural descriptions, not to be interpreted as bearing a direct relation with any computational "steps" in the brain.

This specific interpretation of formalism in generative grammar is not logically necessary, nor is it by any means universal. Chomsky (1995) best states this point of view when, in his criteria for *descriptive* adequacy of grammars characterizing linguistic competence, he includes the criterion that the grammars yield psychologically correct structural descriptions (representations)—not merely the the (computational) criterion that they pick out all and only the grammatical sentences of the language. He does not, however, require that a grammar give us any notion at all of how parsing is done on-line (an algorithm: see pp. 17–18). Some researchers, however, have interpreted the derivational steps one could go through using a phrase-structure grammar in to derive a sentence as making empirically testable algorithmic assertions about how structure is actually built. For example, Miller and Chomsky (1963) proposed that psycholinguistic criteria for evaluating a syntactic grammar could be developed, under the assumption of a "transparent" parsing, with a list of rewrite rules stored explicitly in memory, and parses by executing steps corresponding to these rules. See Berwick and Weinberg (1984) and Phillips (1996) for further discussion. On the other hand, some authors have suggested that the only reasonable place to situate formal grammars is at the purely computational level, specifying the function computed by the mind, and nothing more (Matthews, 1991). Under this view, two different theories that predict the same set of possible words or sentences would be equivalent. All the rules, representations, and derivations would be purely instrumental. This is not a standard view however. Derivations are typically not intended as cognitively real *steps,* but the representations they yield (including, presumably, intermediate steps in serial derivations) are thought to have empirical reality (apart from their indirect consquence that they help the grammar to pick out the right set of grammatical strings).

Few linguists would suggest that opening the brain and looking around would be sufficient to come to a satisfactory solution to the implementational mapping problem with respect to a given linguistic theory, and few would assume that the problem of knowing what to look for has some simple solution from neuroscience (it does not). The same holds if we wish to exploit neural networks to model learning of grammatical knowledge: we must somehow deal with the problem of knowing how to interpret what is learned as a grammar of some kind.

The implementational mapping problem in connectionism

Given the enthusiasm about the analogues between neural networks an brains in the introductory chapters of the original connectionist manifesto (Rumelhart and McClelland, 1986b), one could be forgiven for thinking that, in these early days, the mere existence of a neural-network model of some cognitive task might have been interpreted as having solved the scientific problem of understanding the relevant human brain system, even if no actual brain were ever compared with the model. Great enthusiasm about standard neural networks as veridical brain models is no longer common. Despite the fact that they have their origins in brain models, and despite the fact that they are typically made up of many binary response functions—which can be said to "spike," just like many neurons do—they are today largely seen as powerful families of statistical models with many parameters, rather than attempts at faithful models of physical brains. Typical modern feed-forward neural networks differ in too many ways from real brains to be able to allow us to automatically draw the inference that there is, or even could plausibly be, something comparable going on in the brain system responsible for some aspect of cognition, just because we are able to construct a neural network model.

Neural networks are not brains, but they are complicated and opaque systems. The sheer number of parameters and their sometimes very indirect and complicated relations with the inputs and outputs of the task on which they are trained means that any given neural network, if it is to be truly understood, requires an analysis as a complex system in and of itself. The Marrian framework can provide an approach to doing this: if the parameters of the system are seen as the "physical" reality of the system, then a correct algorithmic-representational description of the system would allow researchers to correctly reason about the information coded in the system and how it passes through various layers of the network. Such a description must be deduced, like that of any other opaque complex system.

The state of the art in such analysis of networks is somewhat limited; this is unfortunate, because among the motivations for wanting to construct theories of what networks are doing are some practical and pressing ones. New regulations in the European Union require that automated decisions taken on the basis of user-provided information be explained to the end user (Goodman and Flaxman, 2016). This is not currently possible for any satisfying sense of "explanation"—current attempts are typically limited to telling the user that a decision (to show an advertisement, for example) was taken "on the basis of" a laundry list of relevant factors. Reporting which features have the most importance for a given classification in a feed-forward neural network is a well-explored problem (Erhan et al., 2009; Springenberg et al., 2014), but recovering a simple description of the network's "chain of reasoning" is not. And the problem is not merely one for end-users. Neural network models are increasingly used in mission-critical systems where developers and system testers must be able to reason in some way about the network's behaviour in order to ensure that the system is safe. From this perspective, neural networks can be seen as complex, automatically generated computer programs for which no human-readable "source code" is available—for which such source code must be inferred if the system is to be debugged. At present, we are far from having good, general-purpose methods for doing so.

There is another reason, more immediately relevant to a cognitive scientist, to be interested in mapping neural network models to high-level formal "theories" describing their operation. Although standard neural networks are not faithful biological models, it is substantially *easier* to imagine turning them into serious and realistic brain models than to imagine starting from scratch. There is certainly something brain-like about typical neural networks—they are made up of many binary response functions, and brain systems contain many spiking neurons; there are even plenty of encouraging results using out-of-the-box neural networks to predict real neural activation, giving at least suggestive hope that the gap in the mechanisms could perhaps be bridged (see Kietzmann et al. 2017). Thus, an effort to map a neural network's internal representations to those of some formal theory—and the consequent step of comparing different hypotheses about how to best characterize a network's representations—could be a first step towards developing characterizations of what various theoretical proposals might look like if implemented neurally—helping us to then know what to look for. Whether or not this proves to be true, I suspect strongly that the simple project of engaging with the implementational mapping problem will prove useful for the project of understanding real brains—even if the complex systems we initially analyse are completely artificial—in the same way that developing grammatical formalisms for artificial formal languages was a crucial step in the development of linguistic theories which continues to be a fruitful source of mathematical insight about human languages today.

Determining whether, or to what degree, an algorithmic–representational theory really reflects how a neural network is operating can be seen as a problem of fitting a model to data—except that, in this case, the "data" consists of internal information about how a network treats specifically selected inputs, and the model provides some set of expectations about how it *should* treat this inputs. Model fitting is best seen as a comparative exercise of trying to select the comparatively best model for a given set of data. We can think of two fundamental problems, therefore, in the assessment of how a neural network should be interpreted: can a given network be seen as implementing, or approximating, the algorithmic-representational theory X, or is network Y a better model? Is a given algorithmic-representational theory implemented or approximated by network M, or is network N closer to the predictions of the theory?

While this idea of "decoding" the action of connectionist networks in terms of some interpretable "model" of their behaviour is quite old, the theoretical tools for doing so are still far from being well-developed enough to be commonly applied to real networks. The paper by Pinker and Prince (1988), responding to the connectionist past-tense inflector of Rumelhart and McClelland (1986a), did not provide any reusable methodological tools for understanding other neural network models; but it did provide a long exegesis of the behaviour of a single network, providing proof that such networks could be "understood," and did not need to remain black boxes. Properly within the realm of neural network research, as discussed above, a number of techniques have been developed for assessing the relative importance of the different input features in determining the output; relatedly, starting in the early 1990s, a number of "rule extractor" techniques were developed that could be used to turn the parameters of neural networks into sequences of rules operating on the (possibly continuous) input space of the network (sometimes yielding only approximations to the network's behaviour). However, none of these techniques are applicable to the problem of extracting sequences of exact rules

from the types of network architectures in common usage today (Taylor and Darrah, 2005; Özbakır et al., 2010; Augasta and Kathirvalavakumar, 2012). Furthermore, none provides useful general tools for addressing the two fundamental problems just presented.

Despite the relative methodological infancy of the field, the problem of decoding neural network information processing has seen a sharp uptick in interest in recent years, probably due in some proportion to the practical need to interpret working systems, but also at least as much (and likely more so) to the "analogy" result of Mikolov et al. (2013). The authors demonstrated that the representation of *king*, learned by a neural network solely on the basis of sequential relations in natural text corpora, stood in the same geometric relation to the representation of *queen* as *man* stood to *woman*, and, more generally, that representations in the system approximated an implicit semantic feature structure. The methodology used to determine this provides a method for assessing, given a specific hypothesized set of binary featural contrasts, whether one network approximates them better than another. The method has been applied, with some variations, to a host of other types of representations since (Gladkova et al., 2016; Dunbar et al., 2015; Linzen et al., 2016b; Chaabouni et al., 2017). The original method has been criticized as giving erroneous results in various cases, and for making narrow and unmotivated assumptions about the form of the mapping between the network's representations and the hypothesized feature system (Levy and Goldberg, 2014; Linzen, 2016), but it provided both a compelling result and a reminder of a compelling idea—old, but under-sold: there are ways of interpreting neural network representations as respecting the structure of some abstract, and understandable, theory.

Tensor product representations (Smolensky, 1988b) represent a more general framework for stating mapping hypotheses between neural network representations and formal representational theories, which allow for mappings between network weights and complex hierarchical representations. However, up to now, published work in this framework has not directly addressed the two fundamental decoding problems. For example, Palangi et al. (2017), cited by Pater, presents a network which is specifically trained to be interpretable by *imposing* a specific mapping to a linguistic representation. A different line of research is represented by the method of Kriegeskorte et al. (2008) for assessing isomorphisms of two representational systems. This method is, in principle, applicable to the implementational mapping problem, but it has almost exclusively been used for comparing neural network representations against equally opaque data from brain imaging (seen as "representations" in a broad sense).

Conclusion

Neural networks are complicated and opaque, and must be analyzed as such. Pater insists on the immediate importance of a research program that goes back to the early days of connectionism (Fodor and Pylyshyn, 1988; Smolensky, 1988a; McCloskey, 1991): neural networks can and should be seen as implementations of (approximations to) formal theories of linguistic knowledge. This is indeed a promising research direction whose time has come. However, in the spirit of Marr's approach to understanding the brain as a complex system, before an interdisciplinary research program of the kind envisioned by Pater can be made real, a very concrete, broad and well-thought-out set of formal methodologies for posing the question of how well a particular neural network aligns with a linguistic theory, stated at Marr's algorithmic-representational level toolkit, is needed. Without careful reflection on what it would mean for a grammatical hypothesis to be "instantiated" in a neural network, and ways to understand what kind of alternative representations a network model is proposing to us, we will be lost in this endeavour.

The above discussion should not be taken to suggest that there is a general, universal solution to the implementational mapping problem. Take the second fundamental decoding problem, giving relative scores to two different networks to assess which one is a closer approximation to a given representational system. Mikolov et al's "analogy" approach, and related approaches to the problem of assessing coherence of a network with binary feature representations, work by assessing whether minimal feature oppositions, such as those between *king* and *queen*, or *man* and *woman* (differing minimally in semantic gender), can be captured by doing arithmetic addition and subtraction operations (does *king* - *queen* + *man* equal *woman*?). This assumes a very specific type of implementational mapping, wherein the structure of the network's representational space is necessarily one in which addition of the vectors representing two words is the network's equivalent of feature composition. This may be reasonable in specific types of networks—and it may be true—but it may also be inappropriate: a network might "contain" binary semantic features in some different sense, using a structure based on something other than addition. Indeed, we already know that there is no single, universal way of "interpreting" real neurons: gerbils and chickens' neural systems for sound localisation in the horizontal plane in front of the body can be described in the same algorithmic-representational terms—calculating inter-aural time differences—but this information is coded in neural firing in a fundamentally different way across the two species (see Ashida and Carr 2011). Any attempt to "find" inter-aural time differences in gerbils under the expectation that it should be encoded

in the same way as in chickens would lead to a negative result (they are coded using neural firing rate in mammals, but not in birds). This would lead to the false conclusion that the two species' brains perform fundamentally different operations, while, in fact, they perform they same operations in different ways. The problem is not insurmountable, but simply needs to be taken into account: any assessement of the congruence between an opaque (neural network) representation and a high-level formal theory is necessarily and inextricably bundled with the assessment of the validity of the mapping hypothesis.

It should also be pointed out that there are several related lines of research which are important, but which do not represent formal tools for evaluating formalism–network mappings in the way I sketch here. First, as discussed by Pater, violable constraint-based grammar formalisms have their origins in the insight that certain classes of neural networks can be also be formalized as sets of weighted constraints (Smolensky, 1986). The import of this is that the problem of mapping to a neural network has a solution for a specific type of formalism and a very specific type of network architecture; it is not in and of itself an attempt to find ways of analysing other types of networks or other types of formalisms. Second, much recent literature has attempted to build linguistically informed tests assessing whether learned systems form correct linguistic generalizations—for example, the goal of the paper by Linzen et al. (2016a) cited by Pater is to construct a set of grammaticality judgment tests to assess whether networks trained on text form correct generalizations about agreement in English (see also Gulordava et al. 2018 for an updated, and rather different, result). While this paper can be seen as assessing whether the network's knowledge of English is consistent with hierarchically "structure"-dependent generalizations, it does not directly demonstrate the presence of hierarchically structured representations in the network. This kind of careful, linguistically informed *extrinsic,* or behavioural, evaluation is also critical to the interdisciplinary project proposed by Pater. But it is not the same as the equally critical project of understanding how we can interpret networks using abstract formalisms and, conversely, how we can ground abstract formalisms in neural network implementations.

References

Ashida, G. and Carr, C. E. (2011). Sound localization: Jeffress and beyond. *Current opinion in neurobiology*, 21(5):745–751.

Augasta, M. G. and Kathirvalavakumar, T. (2012). Reverse engineering the neural networks for rule extraction in classification problems. *Neural processing letters*, 35(2):131–150.

Berwick, R. and Weinberg, A. (1984). *The Grammatical Basis of Linguistic Performance: Language Use and Language Acquisition*. MIT Press, Cambridge, MA.

Chaabouni, R., Dunbar, E., Zeghidour, N., and Dupoux, E. (2017). Learning weakly supervised multimodal phoneme embeddings. In *Proceedings of INTERSPEECH 2017*.

Chomsky, N. (1995). *The Minimalist Program*. MIT Press, Cambridge, MA.

Dunbar, E., Synnaeve, G., and Dupoux, E. (2015). Quantitative methods for comparing featural representations. In *Proceedings of the 18th International Congress of Phonetic Sciences*.

Erhan, D., Bengio, Y., Courville, A., and Vincent, P. (2009). Visualizing higher-layer features of a deep network. Technical Report 1341, Université de Montréal.

Fodor, J. A. and Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

Gladkova, A., Drozd, A., and Matsuoka, S. (2016). Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *Proceedings of the NAACL Student Research Workshop*, pages 8–15.

Goodman, B. and Flaxman, S. (2016). European Union regulations on algorithmic decision-making and a "right to explanation". In *ICML Workshop on Human Interpretability in Machine Learning*.

Gulordava, K., Bojanowski, P., Grave, E., Linzen, T., and Baroni, M. (2018). Colorless green recurrent networks dream hierarchically. *arXiv preprint arXiv:1803.11138*.

Kietzmann, T., McClure, P., and Kriegeskorte, N. (2017). Deep neural networks in computational neuroscience. In *Oxford Research Encyclopedia of Neuroscience (online)*.

Kriegeskorte, N., Mur, M., and Bandettini, P. A. (2008). Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2:4.

Levy, O. and Goldberg, Y. (2014). Linguistic regularities in sparse and explicit word representations. In *Proceedings of the eighteenth conference on computational natural language learning*, pages 171–180.

Linzen, T. (2016). Issues in evaluating semantic spaces using word analogies. In *The First Workshop on Evaluating Vector Space Representations for NLP*.

Linzen, T., Dupoux, E., and Goldberg, Y. (2016a). Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Linzen, T., Dupoux, E., and Spector, B. (2016b). Quantificational features in distributional word representations. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*.

Marr, D. (1982). *Vision: A computational investigation into the human representation and processing of visual information*. MIT Press, Cambridge, MA.

Matthews, R. J. (1991). Psychological reality of grammars. In Kasher, A., editor, *The Chomskyan Turn*, pages 182–200. Blackwell, Oxford.

McCloskey, M. (1991). Networks and theories: The place of connectionism in cognitive science. *Psychological science*, 2(6):387–395.

Mendelson, E. (1997). *Introduction to Mathematical Logic*. Chapman and Hall, New York, fourth edition edition.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Miller, G. and Chomsky, N. (1963). Finitary models of language users. In Luce, D., editor, *Handbook of Mathematical Psychology*, pages 419–491. John Wiley and Sons.

Özbakır, L., Baykasoğlu, A., and Kulluk, S. (2010). A soft computing-based approach for integrated training and rule extraction from artificial neural networks: Difaconn-miner. *Applied Soft Computing*, 10(1):304–317.

Palangi, H., Smolensky, P., He, X., and Deng, L. (2017). Deep learning of grammatically-interpretable representations through question-answering. *arXiv preprint arXiv:1705.08432*.

Phillips, C. (1996). *Order and structure*. PhD thesis, MIT.

Pinker, S. and Prince, A. (1988). On language and connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition*, 28(1-2):73–193.

Rumelhart, D. E. and McClelland, J. L. (1986a). On learning the past tenses of english verbs. In James L. McClelland, D. E. R. and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 2: Psychological and biological models*. Bradford Books/MIT Press, Cambridge, MA.

Rumelhart, D. E. and McClelland, J. L. (1986b). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*. MIT Press, Cambridge, MA.

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In James L. McClelland, D. E. R. and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundationa*, pages 194–281. Bradford Books/MIT Press, Cambridge, MA.

Smolensky, P. (1988a). The constituent structure of connectionist mental states: A reply to fodor and pylyshyn. *The Southern Journal of Philosophy*, 26(S1):137–161.

Smolensky, P. (1988b). On the proper treatment of connectionism. *The Behavioral and Brain Sciences*, 11.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. In *International Conference on Learning Representations*.

Taylor, B. J. and Darrah, M. A. (2005). Rule extraction as a formal method for the verification and validation of neural networks. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 5, pages 2915–2920. IEEE.