Generative grammar, neural networks, and the implementational mapping problem

Ewan Dunbar
Laboratoire de Linguistique Formelle
(CNRS - Université Paris Diderot - Sorbonne Paris Cité)
ewan.dunbar@univ-paris-diderot.fr

To appear in Language *March, 2019*
Commentary on Joe Pater, *Generative linguistics and neural networks at 60: foundation, friction, and fusion* (link to original paper)

Abstract

The target article proposes to use neural networks to model learning within existing grammatical frameworks. This is easier said than done. There is a fundamental gap to be bridged which does not receive attention in the article: how can we use neural networks to examine whether it is possible to learn some linguistic representation (a tree, for example), when, after learning is finished, we cannot even tell if this is the type of representation that has been learned (all we see is a sequence of numbers)? Drawing a correspondence between an abstract linguistic representational system and an opaque parameter vector which can (or perhaps cannot) be seen as an instance of such a representation is an *implementational mapping problem.* Rather than relying on existing frameworks that propose partial solutions to this problem, such as harmonic grammar, I suggest that fusional research of the kind proposed needs to directly address how to "find" linguistic representations in neural network representations.[1]

Keywords: neural networks; grammatical formalisms; cognitive science; implementational mapping; generative grammar

1. INTRODUCTION

On the same list that ranked *Syntactic Structures* as the number one most influential twentieth-century work in cognitive science, the number two work is David Marr's *Vision* (Marr 1982). Marr proposed that the only way to achieve an understanding of complex information processing systems (like those found in the brain) is to simultaneously analyze them at three levels of analysis: a **computational** level, where one formally defines the problem the system is solving, specifying which inputs must map to which outputs; 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; and an **implementational** level, which details the physical system itself. 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 neural network research immediately runs up against the implementational mapping problem. Neural network models—while not brains, or even brain models—are complex systems whose behaviour cannot be understood just by inspecting their internal state. On the other hand, representations proposed in linguistic theories are designed so that human researchers can write and read them—hypotheses of the kind that Marr had in mind for his intermediate level, explicit enough to be programmed on a computer, yet high-level enough to be understood. They are not directly comparable with the internal states of neural networks, which are simply long sequences of numerical parameters. "When neural network modelling is *integrated with* grammatical formalisms," writes Pater, "we may be able to go further in assessing the extent to which *grammatical representations* can be learned from experience" (emphasis mine). The sketch Pater outlines in ***Fusion*** is missing a component critical to an integration of this kind. If the network is seen as the implementation, then a fundamental part of the work of "integration" with grammatical formalisms consists in solving the mapping problem.

In principle, feedforward networks can implement any grammar, or any formal representation, or approximate it with arbitrary precision—with the basic units of representation implemented as vectors in some numerical parameter space, and with operations manipulating these representations corresponding to information flowing through the network, changing its state (Hornik 1991; Leshno *et al.* 1993; Lin *et al.* 2017). This makes them attractive for simulating grammatical learning. But doing so first requires answering some fundamental questions:

- Given a trained network, does it make use of representations consistent with a linguistic theory T? (Can we, for example, see its representations as trees: Smolensky 1988b; Pollack 1988?) Or is such an interpretation provably untenable?
- …or has the network *approximately* arrived at a solution within the scope of T—for example, one for which a subset of the representations can be seen as trees?
- Can all possible parameter settings learnable by a given class of network N (constrained to have a certain architecture, with a certain type of units) be seen as licit grammars in theory T? In other words, is N constrained to learn within theory T (or come close)?

- Conversely, can all grammars in T be implemented as networks in T?

Only when we know how to answer these questions can we address the question the target article hopes to be able to answer by fusing the two research programs: given a class of networks N that has the capacity to go *beyond* some theory of possible grammars T, is an arbitrary network of this class guaranteed to learn a grammar in T if we give it realistic data? Or does it require special evidence, inaccessible to the child, in order to be constrained to grammars in T? This would allow us to conclude something useful: that the learner must have mechanisms to constrain it to T (N is too broad), or that the theory of learning from data is wrong, or that the theory of grammar is wrong (and should maybe allow the alternative solutions available within N). But, to arrive at these answers, we must have an implementational mapping between network and formalism.

2. THE IMPLEMENTATIONAL MAPPING PROBLEM

When a human brain is cut open, trees do not come fanning out, but this does not this mean that syntactic theory is wrong. This kind of observation is the essence of Marr's methodological program. Even if we had access to physical state of every cell in the brain, we would need to do work to understand whether their activity could be seen as building trees, or something rather different. Before we can validate an abstract theory of how the system might work, we need some systematic theory of how the abstract elements and operations map to the physical implementation. Equally, if we wish to simulate learning of generative grammars with neural networks as "hardware," particularly in the case where we do not force the networks to learn specific kinds of representations, we need some system for linking an abstract formal theory to the networks' representations. This implies making the formal theory explicit—and noting which elements of it are there to describe **how** the system does its work, rather than just characterizing **what** it does—and then articulating **how we would recognize** what the network is doing.

To see the general problem, take the artificial example of addition of natural numbers. Addition is a function that applies to two natural numbers, and results in some other natural number. Physical implementations of addition include the proper use of an abacus, standard pencil and paper column-based arithmetic with carrying, or a spreadsheet taking inputs with a keyboard and giving outputs on a computer monitor. For each of these very different systems, we can say that addition is **what** they do. We can only say this with certainty, however, once we characterize exactly what addition is. A computational-level description of a system implementing addition 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? One set of axioms for arithmetic is due to Giuseppe Peano (given in Mendelson 1997), and asserts, for example, that the set of manipulated elements must contain a special element called **0**, that there be a function called $successor(x)$ which always yields an element distinct from x, that $add(x, y)$ yield x when $y = \mathbf{0}$—and, in all other cases, $add(x, successor(y)) = successor(add(x, y))$—along with several other details. When these axioms are put together, they have as logical consequences that the inputs must be a system equivalent to natural numbers, and specify uniquely what the output of any addition must be.

The physical system itself will be quite far removed from any computational-level description. A partial understanding of how the system works on a purely physical level might be enough to

verify whether a hypothesis about *what* the system is doing is correct, but Marr proposes that, to meaningfully link between a computational-level description and a physical implementation, a detailed, abstract hypothesis is necessary, which goes beyond describing *what* a system does, to making a claim about *how* it does it. Marr calls this an "algorithmic–representational" description. Taking the example of addition: how are numbers represented? Possibilities might be decimal ($successor(4) = 5$), binary ($successor(10) = 11$), unary ($successor(||||) = |||||$), or Roman numerals ($successor(\text{IV}) = \text{V}$). Representational systems can be spelled out in formal detail. A unary encoding, for example, contains a basic element that we write |, is such that every representation consists only of some number of |'s, and (drawing the link to the computational-level description) is such that longer strings always represent larger numbers and conversely (contrast this with a decimal system, in which "41" is not longer than "14"). In concert with formal definitions of "sequence" and of "longer," this is enough to specify that a representation is unary. Then, we need an algorithm: a set of implementation-neutral steps for manipulating representations which yield results consistent with the computational-level description of the system. The familiar column-wise addition with carrying that we learn in school can be spelled out explicitly, for example.[1] The characterization of a representational system and an algorithm must provably give results consistent with the computational-level constraints.

Thus, given a hypothesis about *what* the system does, and *how*, the implementational mapping is a theory of *how I recognize* what the system is doing when I observe it operating.[2] If you have been to school in the West, you can probably quickly tell when addition is what's being done in a sequence of scribbles on paper. This background allows you to map numbers (the basic elements which form the inputs and outputs of addition) onto marks on paper, and to recognize the steps of a method for adding numbers together using these marks. Other systems implementing addition might be completely opaque—impossible to *recognize* as implementing addition—even to an adult with a solid knowledge of how addition works: 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. Some hypothesis is therefore needed about how the elements of the algorithmic–representational description map to the physical reality. A system implementing unary encoding (fingers, sticks, coins) would not need to contain actual physical lines written |, but it would need to at least be coherent with the representational constraint that there be a one-to-one mapping between the "size" of the representation and the number represented—the relevant "size" would not need to be the physical volume, but there would need to be some such coherently definable physical

[1] The choice of algorithm and of representation are not independent: column-wise addition will work with decimal, binary, or (with some contortions) unary representations; it will not work with Roman numerals. And, although it can be made to 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.

[2] This should be kept distinct from a theory of the physical system itself. A complete understanding of the system implies having such a theory—and constructing an implementational mapping does too—but one could come to fully understand where the beads in an abacus can go without understanding why they go there.

quantity—and this would not exist for all systems: it exists in the addition-of-pure-tones system (the frequency) but not in the abacus (there is no simple physical measure by which the beads representing 14 on a base ten abacus have ”less” of it than the beads representing 41). The steps in the algorithm, too, must be mapped to changes of state in the physical system, in some way. Drawing these links explicitly is what is meant by an implementational mapping.

Marr’s approach thus gives us two insights that are critical to Pater’s proposed project of fusion. First, “how” is not “what.” The assertions that are meant to be taken at the algorithmic–representational level must be cleanly distinguished from the computational-level description. The “what” theory of addition contains “elements” and “operations,” too, but the point of having two levels of description is that only the algorithmic–representational description needs to correspond to the physical reality. For example, the function $successor(x)$ exists at the computational level only: there does not need to be a basic physical operation “add one” in order for a system to be correctly described by these axioms. But a system implementing the algorithm of adding in columns *would* need to have a physical change of state corresponding to carrying. The implication of this is that observing that a network has the same behaviour as what is predicted by some linguistic analysis is not the same as asking whether it can learn grammatical representations. In this context, it is worth explicitly drawing the comparison between Marr’s program and the reflection about the cognitive reality of grammars that happens in generative linguistics. We usually tell students that the series of steps carried out *on paper* to derive a sentence, or a phonological surface form, are not supposed to be a theory of the steps carried out *in the mind*—no theory of the *algorithm* is implied by the theory of the grammar—but that the representations (trees, feature bundles, and so on) *are* supposed to be cognitively intepreted; the derivational steps are interpreted as a means of stating which representations are licit structural descriptions. Chomsky (1995) best articulates this interpretation: in his criteria for *descriptive* adequacy of grammars characterizing competence, he includes the criterion that the grammars yield psychologically correct structural descriptions (representations)—not merely 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 of how parsing is done on-line (an algorithm).[3] Under this view, the “operation” $merge(\cdot, \cdot)$ is an implicational relation in a system of representations (if X and Y are representations, then so is $merge(X, Y)$), not a theory of a real-time processing step. The theory, under this view, is not purely computational-level (this would only be concerned with defining the correct set of utterances), nor a complete theory at the algorithmic–representational level (because this would also include proposals of algorithms for generation and recognition). On the other hand, some researchers have interpreted derivational steps as making empirically testable assertions about how structure is built up in time (Miller & Chomsky 1963; Berwick & Weinberg 1984; Phillips 1996), while other authors have suggested that the only reasonable place to situate formal grammars is at the computational level, specifying the function computed by the mind, and nothing more (Matthews 1991). Under that view, two theories that predict the same set of possible words, sentences, or sound–meaning pairs, would be equivalent; their rules, representations, and derivations are purely instrumental. Regardless of how we use generative grammars, the spelled-out reflection is what is important. For doing everyday linguistics, it may not be all that important to regularly draw up the list of which elements of the formal theory are meant to be interpreted as “real” and which not.

[3] See pp. 17–18 of Chomsky 1995

This exercise is critical for doing learning in neural networks to see if they learn "the same thing" as we have proposed in some theory, because it serves to explicitly delimit the success criterion.

Second, of course, Marr's program asks us to spell out the details of the "how do I recognize it" theory, making it clear that this is not trivial. The opacity of neural network representations poses a central challenge to any attempt to use them to implement grammars.

3. IMPLEMENTATIONAL MAPPING AND THE PROJECT OF FUSION

The target article points to two existing points of convergence between neural network research and generative grammar. First, harmonic grammar, a fully-developed linguistic theoretical framework based on constraint interaction, and an existing connectionist–generativist "fusion"; second, a set of studies testing recurrent neural networks trained on corpora to see whether they yield human-like judgments. The first is an implementational mapping for grammatical computations which is problematic in that it is very limited; the second does not evaluate anything at the algorithmic–representational level.

Harmonic grammar was born out of a desire to come up with an abstract formal theory for understanding the operations of neural networks: an implementational mapping, with neural networks seen as the implementation of an abstract formal theory. The theory could then be used by linguists, with the knowledge that there exists a way of translating linguistic analyses into learned network parameters. The system rests on two formal mechanisms. The first, *tensor product representations*, provides a way of mapping between formal representations and numerical vectors. The second, *harmony theory*, provides a way of formally translating between soft constraint optimization and certain kinds of transformations that neural networks can do over tensor product representations.

Harmony theory maps a network with two layers to a grammatical computation. The first layer is a representation vector, linked to the second layer, which might be called a *constraint satisfaction vector* (Smolensky (1986); Legendre *et al.* (1990a); Legendre *et al.* (1990b)). Each unit in the second layer represents a different constraint, and has a value, calculated from the first layer in the usual way in multi-layer perceptrons: by taking a linear combination of values and passing them through an activation function. The constraint units in turn have weights, which (by again applying the standard linear calculation) allow us to arrive at a value ("harmony") interpretable as a degree of acceptability. The constraint satisfaction vector is interpretable and manipulable by a linguist, giving a simple implementational mapping for a part of the grammatical calculation. By "calculation" is meant the deduction of unknown structure: calculating interpretations from surface forms (via phonological and morpho-syntactic structure), or calculating surface pronunciations in the other direction. However, harmony theory requires a rather uncommon type of network. The network needs to do computation by "optimization," rather than using a series of feed-forward multiplication/addition steps. Both the known and unknown representations must form part of the first layer, in which case, grammatical computation can be done by holding the known elements constant, and doing optimization over the unknown representation so as to maximize harmony. There is a class of networks (including Hopfield nets and restricted Boltzmann machines) with this "fill in the blank" architecture, in which a single layer contains both observed and hidden quantities, but such architectures are very rare in practice. In reality, virtually any representation one can find in a modern neural network is the result of doing a series of linear transformations followed by various types of non-linearities,

using weights that are learned and fixed once during training. Harmony theory does not yield an implementational mapping to a constraint-based grammar in such a case.[4]

This major architectural difference means that harmonic grammar per se currently has little to offer in terms of practical fusion. Indeed, as Pater points out, in practice, HG learning is almost always modelled in the case "when the structure of the learning data is supplied in whole—when all the constraint violations of each learning datum are known"—that is to say, when only the final constraint layer is being modelled, and the weights that make the link between representation and constraint violations are abstracted away from. In this case, the model has been simplified past the point where it can even be called a neural network. It is no longer a multilayer perceptron, but simply a single-layer perceptron—a type of generalized linear model—which is what allows for the use of the wide range of off-the-shelf learning algorithms alluded to in target article. Omitting the representation from consideration means that little rests of the big, fusional questions put forward in the introduction. All that is being learned by the "network" is a set of weights on constraints; no questions about the learnability of representations, or the constraints themselves, are being addressed by a neural network.

Tensor product representations, a family of implementational mappings for representations (rather than for grammars and grammatical computation), are more generally applicable. After decomposing the representational system axiomatically into its essential elements ("roles"), and the possible values these elements can take on ("fillers"), the mapping hypothesis is that fillers and roles each correspond to possible vectors in the representational space of the network; that fillers are matched to roles using a simple algebraic multiplication operation; and that the resulting elements are combined using vector addition. Complex hierarchical structures can be represented in this way, because a filler vector need not be an atomic element of the representational system (Smolensky 1986; Smolensky 1988a; Smolensky *et al.* 2014).

This implementational mapping scheme is one of the things that ensures the smooth operation of harmony theory. The grammars which come out of applying the isomorphism just discussed to two-layer neural networks are only workable if the input layer is coded in a "localist" fashion, that is, with individual nodes or sets of nodes corresponding to individual elements of the representation. Tensor product representations, on the other hand, can yield "distributed" representations. In distributed representations, individual nodes are not identifiable with representational elements: in the worst case, all nodes may need to be examined in order to decode any part of the representation. The existence of the tensor product mapping scheme means that one can always find a localist re-coding for any given representation. However, tensor product representations are completely independent of harmony theory, forming a broad family of implementational mapping theories for representations. Pater's commentary discusses a case of a question–answer system in which a tensor product structure is *enforced* during network training (Palangi *et al.* 2017), but with no interpretation of the final weights as a harmonic grammar. One

[4] An additional, and important, part of the harmonic grammar framework, is the idea that traditional grammatical representations, which are discrete, are approximations of real linguistic knowledge, which can have states "in-between" two discrete representations; see Smolensky & Goldrick (2016), for example. I gloss over this part of the discussion, as it is not immediately relevant to understanding the relevance of the implementational mapping problem. This is one sense in which a network can fail to contain abstract linguistic representations, but can contain representations which are "close," as alluded to in the introduction.

could easily imagine going beyond enforcing structure, to probing trained networks to understand whether they contained a tensor product encoding of some grammatical representational system—in order to attempt to address the fundamental questions outlined in the introduction.[5]

The second type of fusional research cited is a set of studies investigating whether recurrent neural network language models trained on corpora have human-like judgments for agreement (Linzen *et al.* 2016a). I point out here only that these studies, while fundamentally important, do not permit us to draw the kind of conclusions outlined in the introduction. Because the focus is restricted to whether the networks tested match human judgments—essentially, whether they generate the correct set of strings, plus some associated gradient degree of acceptability/set membership—these studies can inevitably only tell us about the accuracy of the models at the computational level.

To make this point more clearly, consider the recent study of Gulordava *et al.* (2018), a follow-up to the tests of agreement in RNNs cited in the target article, using a different trained model (and improving the tests). In contrast to the model of Linzen et al., (which the authors also re-test with their new items), the paper finds that the new RNNs pass the agreement tests, even at long distances. The authors leave the question open as to what the crucial difference between the two models is that allows one to show human-like behaviour, while the other fails. It is a critical question, in light of the fact that neither model is trained with any explicit bias for hierarchical structure. But it is *not* the same as the question the target article raises in this context: "assessing the extent to which *grammatical representations* can be learned from experience." *That* question relies critically on knowing something not only about *what* the system does but *how* it does it. The fact that the system shows the correct behaviour does not have any straightforward implication about the nature of the representations it is using to do so. (Chomsky's (1975) informal reflections about the "structure"-dependence of agreement merely assert that hierarchical structure serves as the basis for a *better* hypothesis than a purely linear rule, not that *no* alternative representation could support the correct mapping; in this regard, this early discussion of the learning problem posed by agreement is representative of later re-assertions of it.) To see whether the better-performing system has really learned to code something we would call syntactic hierarchy, we would need to address the implementational mapping problem for this type of network. We will never be able to use neural networks to assess whether it is possible to learn (a particular type of) grammatical representations from (a particular type of) data, if we do not have a way of assessing whether it has learned those representations.

4. HOW TO PROCEED

Tensor product representations represent one framework for stating mapping hypotheses between neural network representations and formal representational theories, which allow for

[5]One could also imagine attempting to decode higher layers of a feed-forward or recurrent network in such a way as to allow the interpretation of these layers as making up parts of a constraint satisfaction vector, as in harmonic grammar. This would still not permit the complete implementational mapping found in harmony theory, however, as this would imply outputs and hidden structure that are calculated in a fundamentally different way, by optimization, rather than the standard feed-forward mechanisms.

mappings between network weights and complex hierarchical representations. There are other approaches to "decoding" the action of connectionist networks in terms of some interpretable "model" of their behaviour. The paper by Pinker & Prince (1988), responding to the connectionist past-tense inflector of Rumelhart & McClelland (1986), did not provide any reusable methodological tools for understanding other neural network models; it did, however, 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, 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 & Darrah 2005; Özbakır *et al.* 2010; Augasta & Kathirvalavakumar 2012).

One of the reasons that the problem of decoding neural network information processing has seen a sharp uptick in interest in recent years is probably the impact of 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/syntactic feature structure. The methodology used to arrive at this conclusion 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 & Goldberg 2014; Linzen 2016), but it provided both a compelling result and a reminder of a compelling idea: there are ways of interpreting neural network representations as respecting the structure of some abstract, and understandable, theory.

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).

5. CONCLUSION

The target article insists on the immediate importance of a research program that goes back to the early days of connectionism (Fodor & Pylyshyn 1988; Smolensky 1988a; McCloskey 1991): using neural networks to advance, not as an alternative to, abstract linguistic theory. This is indeed a promising research direction whose time has come. However, 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 is need for assessing how well a particular neural

network aligns with a linguistic representational theory. 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. Suppose, for example, that, in follow-up research on how RNNs learn agreement, we wanted to give relative scores to two different networks, to assess which one was coming closer to having learned hierarchical structure, in some well-defined sense. Mikolov et al's "analogy" approach works by assessing whether minimal representational oppositions, such as those between *king* and *queen*, or *man* and *woman* (assuming that these differing minimally in some representation of semantic gender), can be captured by doing arithmetic addition and subtraction operations (does *king* - *queen* + *man* equal *woman*?). Tensor product representations make a similar assumption. Both assume a specific class of implementational mappings wherein the structure of the network's representational space is necessarily one in which addition of two representations is the network's way of composing these representations. This may be reasonable in specific types of networks, but it is not guaranteed to be inappropriate: a network might "contain" binary semantic features in some different sense, using a structure based on something other than addition.

We already know from neuroscience 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 & 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.

The early work on linking neural network representations with more understandable views of them as abstract rules and representations was extremely important, and has advanced far too slowly in the intervening years. Furthermore, it still not always clearly distinguished from the problem of testing whether the networks get the right answer. This lack of centrality shows in the target article, which proposes extensive fusion, but does not isolate implementational mapping as a critical component, or even as a research problem. Given the complexity of the problem, I would suggest that implementational mapping represents, in fact, the bulk of the work of fusion between formal linguistic theory and neural network research.

References

ASHIDA, GO, & CATHERINE E CARR. 2011. Sound localization: Jeffress and beyond. *Current opinion in neurobiology* 21.745–751.

AUGASTA, M GETHSIYAL, & THANGAIRULAPPAN KATHIRVALAVAKUMAR. 2012. Reverse engineering the neural networks for rule extraction in classification problems. *Neural processing letters* 35.131–150.

BERWICK, ROBERT, & AMY WEINBERG. 1984. *The Grammatical Basis of Linguistic Performance: Language Use and Language Acquisition*. Cambridge, MA: MIT Press.

CHAABOUNI, RAHMA, EWAN DUNBAR, NEIL ZEGHIDOUR, & EMMANUEL DUPOUX. 2017. Learning weakly supervised multimodal phoneme embeddings. In *Proceedings of INTERSPEECH 2017*.

CHOMSKY, NOAM. 1975. *Reflections on language*. New York: Pantheon.

——. 1995. *The Minimalist Program*. Cambridge, MA: MIT Press.

DUNBAR, EWAN, GABRIEL SYNNAEVE, & EMMANUEL DUPOUX. 2015. Quantitative methods for comparing featural representations. In *Proceedings of the 18th International Congress of Phonetic Sciences*.

FODOR, JERRY A, & ZENON W PYLYSHYN. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition* 28.3–71.

GLADKOVA, ANNA, ALEKSANDR DROZD, & SATOSHI MATSUOKA. 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*, 8–15.

GULORDAVA, KRISTINA, PIOTR BOJANOWSKI, EDOUARD GRAVE, TAL LINZEN, & MARCO BARONI. 2018. Colorless green recurrent networks dream hierarchically. *arXiv preprint arXiv:1803.11138* .

HORNIK, KURT. 1991. Approximation capabilities of multilayer feedforward networks. *Neural networks* 4.251–257.

KRIEGESKORTE, NIKOLAUS, MARIEKE MUR, & PETER A BANDETTINI. 2008. Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience* 2.4.

LEGENDRE, GÉRALDINE, YOSHIRO MIYATA, & PAUL SMOLENSKY. 1990a. Can connectionism contribute to syntax? Harmonic Grammar, with an application. Technical Report 90–12, Institute of Cognitive Science, University of Colorado at Boulder.

——, YOSHIRO MIYATA, & PAUL SMOLENSKY. 1990b. Harmonic Grammar–A formal multi-level connectionist theory of linguistic well-formedness: Theoretical foundations. Technical Report 90–5, Institute of Cognitive Science, University of Colorado at Boulder.

LESHNO, MOSHE, VLADIMIR YA LIN, ALLAN PINKUS, & SHIMON SCHOCKEN. 1993. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks* 6.861–867.

LEVY, OMER, & YOAV GOLDBERG. 2014. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the eighteenth conference on computational natural language learning*, 171–180.

LIN, HENRY W, MAX TEGMARK, & DAVID ROLNICK. 2017. Why does deep and cheap learning work so well? *Journal of Statistical Physics* 168.1223–1247.

LINZEN, TAL. 2016. Issues in evaluating semantic spaces using word analogies. In *The First Workshop on Evaluating Vector Space Representations for NLP*.

——, EMMANUEL DUPOUX, & YOAV GOLDBERG. 2016a. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics* 4.521–535.

——, EMMANUEL DUPOUX, & BENJAMIN SPECTOR. 2016b. Quantificational features in distributional word representations. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*.

MARR, DAVID. 1982. *Vision: A computational investigation into the human representation and processing of visual information*. Cambridge, MA: MIT Press.

MATTHEWS, ROBERT J. 1991. Psychological reality of grammars. In *The Chomskyan Turn*, ed. by Asa Kasher, 182–200. Oxford: Blackwell.

MCCLOSKEY, MICHAEL. 1991. Networks and theories: The place of connectionism in cognitive science. *Psychological science* 2.387–395.

MENDELSON, ELLIOTT. 1997. *Introduction to Mathematical Logic*. New York: Chapman and Hall, fourth edition edition.

MIKOLOV, TOMAS, KAI CHEN, GREG CORRADO, & JEFFREY DEAN. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

MILLER, GEORGE, & NOAM CHOMSKY. 1963. Finitary models of language users. In *Handbook of Mathematical Psychology*, ed. by David Luce, 419–491. John Wiley and Sons.

ÖZBAKıR, LALE, ADIL BAYKASOĞLU, & SINEM KULLUK. 2010. A soft computing-based approach for integrated training and rule extraction from artificial neural networks: Difaconn-miner. *Applied Soft Computing* 10.304–317.

PALANGI, HAMID, PAUL SMOLENSKY, XIAODONG HE, & LI DENG. 2017. Deep learning of grammatically-interpretable representations through question-answering. *arXiv preprint arXiv:1705.08432* .

PHILLIPS, COLIN, 1996. *Order and structure*. MIT dissertation.

PINKER, STEVEN, & ALAN PRINCE. 1988. On language and connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition* 28.73–193.

POLLACK, JORDAN B. 1988. Recursive auto-associative memory. *Neural Networks* 1.122.

RUMELHART, DAVID E., & JAMES L. MCCLELLAND. 1986. On learning the past tenses of english verbs. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 2: Psychological and biological models*, ed. by James L. McClelland, David E. Rumelhart, & the PDP Research Group. Cambridge, MA: Bradford Books/MIT Press.

SMOLENSKY, PAUL. 1986. Information processing in dynamical systems: Foundations of harmony theory. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundationa*, ed. by James L. McClelland, David E. Rumelhart, & the PDP Research Group, 194–281. Cambridge, MA: Bradford Books/MIT Press.

——. 1988a. The constituent structure of connectionist mental states: A reply to Fodor and Pylyshyn. *The Southern Journal of Philosophy* 26.137–161.

——. 1988b. On the proper treatment of connectionism. *The Behavioral and Brain Sciences* 11.

——, & MATTHEW GOLDRICK. 2016. Gradient symbolic representations in grammar: The case of French liaison. *Rutgers Optimality Archive 1286* .

——, MATTHEW GOLDRICK, & DONALD MATHIS. 2014. Optimization and quantization in gradient symbol systems: a framework for integrating the continuous and the discrete in cognition. *Cognitive science* 38.1102–1138.

TAYLOR, BRIAN J, & MAIJORIE A DARRAH. 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, 2915–2920. IEEE.