Algorithms for Neural Networks

John Morrison (jmorrison@barnard.edu)
Departments of Philosophy and Cognitive Science, Barnard College, Columbia University
3009 Broadway, New York, NY 10027

Abstract

Algorithms play a central role in cognitive science. They help explain how we perceive, speak, remember, navigate, and decide. But it is unclear what it means say that an artificial or biological neural network "implements" an algorithm. The standard proposal is that a neural network implements an algorithm when it has parts corresponding to the steps of the algorithm. But we haven't been able to find many such parts, perhaps because neural networks rarely have them. This has led some to deny that neural networks implement algorithms. As an alternative, I propose that a neural network implements an algorithm in virtue of how quickly it learns alternative input-output mappings. This proposal draws on the learning-to-learn literature in psychology and the transfer learning literature in machine learning. I demonstrate that this proposal productively applies to a number of networks and tasks. It is therefore a promising new framework for integrating cognitive science and neuroscience.

Keywords: Neural Networks; Algorithms; Transfer Learning; Mechanistic Interpretability; Explainable AI

Since the beginning of neuroscience and machine learning, there has been a desire to understand neural networks in terms of algorithms (McCulloch & Pitts, 1943; von Neumann, 1958). It is also a defining commitment of cognitive science that brains, at least, implement algorithms (Newell & Simon, 1972; Marr, 1983). But what does it mean to say that a neural network "implements" an algorithm?

One proposal is that a neural network implements an algorithm when its parts correspond to the steps of the algorithm. As a toy example, consider the input-output mapping in Table 1. At least in principle, a network trained on this mapping might implement the algorithm: Double the input and then add two — 2x+2. Alternatively, it might implement the algorithm: Add one to the input and then double the sum — 2(x+1). According to this proposal, the network implements 2(x+1) only if it has an intermediate part corresponding to x+1, and it implements 2x+2 only if it has an intermediate part corresponding to 2x. This kind of proposal is standard in the philosophical literature about implementation (Piccinini & Maley, 2021). It is also standard in the machine learning literature on mechanistic interpretability (Geiger, Lu, Icard, & Potts, 2021; Elhage et al., 2021; Wang, Variengien, Conmy, Shlegeris, & Steinhardt, 2022; Olsson et al., 2022; Nanda, Chan, Lieberum, Smith, & Steinhardt, 2023). But, at least so far, it hasn't proven productive. For the vast majority of neural networks, we haven't found the relevant parts, perhaps because they don't exist. Consider that, at least so far, the literature on mechanistic interpretability has only provided insight into a handful of circuits on a handful of tasks in a handful of networks.

Table 1: input-output mapping

input	output
1	4
2	6
3	8
4	10

Another proposal is that whether a neural network implements an algorithm depends only on its inputs and outputs, including how it generalizes to new inputs. This is the standard approach in the machine learning literatures on transitive inference and compositionality (De Lillo, Floreano, & Antinucci, 2001; Kay et al., 2023; Lippl, Kay, Jensen, Ferrera, & Abbott, 2023; Hupkes, Dankers, Mul, & Bruni, 2020; Lake & Baroni, 2023). But this proposal does not provide as much insight into neural networks as we would like because it can't distinguish between algorithms like 2x+2 and 2(x+1). In many cases, we would like a way of choosing between algorithms with the same inputs and outputs.

Perhaps due to dissatisfaction with these proposals, some deny that we can understand neural networks in terms of algorithms (P. S. Churchland, 1986; P. M. Churchland, 1989; Horgan & Tienson, 1996; Ramsey, 2007; Lillicrap & Kording, 2019; Cao & Yamins, 2021). But this would be a disappointment, because algorithms are central to cognitive science, and many of us think that cognitive science has a lot to offer neuroscience and machine learning.

My proposal is that neural network implements a neural network because of its learning speed, in particular the number of learning events required to learn alternative input-output mappings. I call this the "learning aptitude definition." Its details require careful introduction. But, to get a feel for it, consider an exceptionally simple artificial network: a fully connected network consisting of one input node, three intermediate layers with eight nodes, four nodes, and eight nodes, and then one output node. Suppose that, after a relatively brief training period, it learns to map 1, 2, 3, and 4 to the outputs specified in Table 1. We'd like to know: Is this network implementing 2x+2, 2(x+1), or has it merely memorized the output for each input? My proposal is that we should consider how long it takes the network to learn the other mappings, including the mappings in Table 2.

At a lower level of description, a network's learning speed is due to its architecture and how its weights are adjusted in response to feedback. But at the higher level of abstraction I'm

Table 2: alternative input-output mappings

input	output
1	6
2	10
3	14
4	18

input	output
1	8
2	12
3	16
4	18

proposing, it is due to the number of adjustments to its algorithm. In particular, if a network implements 2x+2, one adjustment is necessary to learn the mapping on the left (from 2x+2 to 4x+2), while two adjustments are necessary to learn the mapping on the right (from 2x+2 to 4x+4). It will therefore be faster at learning the mapping on the left. In contrast, if a network implements 2(x+1), one adjustment is necessary to learn the mapping on the right (from 2(x+1) to 4(x+1)), while two adjustments are necessary to learn the mapping on the left (from 2(x+1) to 4(x+1/2)). It will therefore be faster at learning the mapping on the right. And if the network just memorized the output for each input, four adjustments are necessary because each output must be individually adjusted. It therefore won't be faster at learning either mapping. We might think of the algorithm's parameters as knobs within the network that must be turned as it learns a new mapping. The more the relevant knobs need to be turned, the longer it takes the network to learn the new mapping. Of course, unlike ordinary knobs, these knobs aren't easy to find because they emerge from the complex, high-dimensional process responsible for adjusting the network's weights. But that doesn't make them any less real.

This proposal builds on an insight originally found in the learning-to-learn literature in psychology (Cormier & Hagman, 1987), and then in the transfer learning literature in machine learning (Thrun & Pratt, 2012; Ruder, 2017). These literatures have a practical focus; their goal is to help people and neural networks learn faster (Vafaeikia, Namdar, & Khalvati, 2020; Zhou et al., 2023). But they are grounded in the insight that it takes less time to learn a new task when some of the the representations and algorithms learned on a previous task can be reused. I propose treating this insight as definitive of what it is for a network to implement an algorithm.

This proposal has a number of surprising consequences. For example, it implies that which algorithm a network implements depends on its learning rule and its training data. In the paper, I argue that these consequences are not as objectionable as they might first appear.

As an illustration, consider two fully connected networks with the architecture described above (1-8-4-8-1) but with different initializations. After the loss for these networks dropped below .1 on the training data generated by $2x+2$, I fine-tuned them on a series of six other mappings (4x+2, 4x+4, 8x+2, 8x+8, 12x+2, 12x+12). Figures 1 and 2 depict their loss during training. The first network was faster at learning the orange mappings (4x+4, 8x+8, 16x+16), while the second network was faster at learning the blue mappings (4x+2, 8x+2, 16x+2). According to the learning aptitude definition, it follows that the first network does not implement 2x+2 but might implement 2(x+1), while the second network does not implement 2(x+1) but might implement 2x+2.

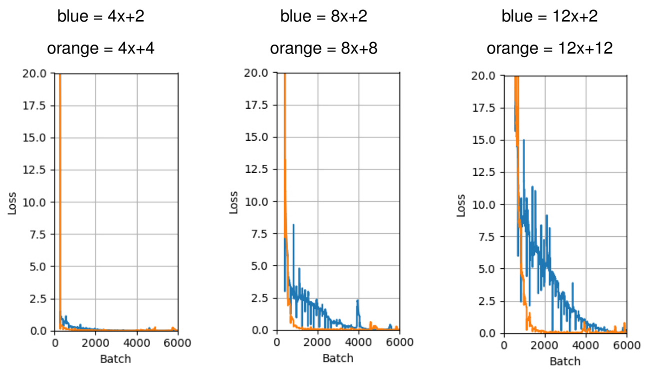

Figure 1: First network trained on 2x+2

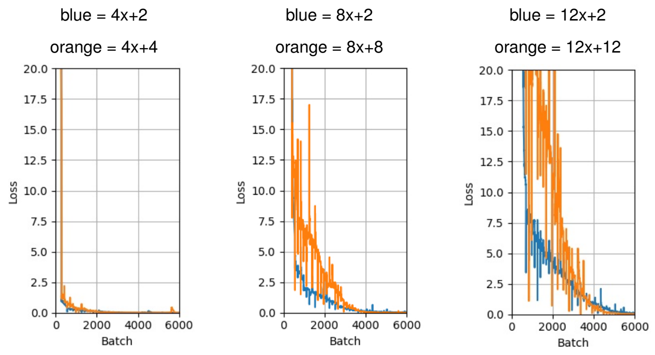

Figure 2: Second network trained on 2x+2

This is just one example. In the paper, I consider other models, tasks, and algorithms, including CNNs trained on ImageNet that implement different classification algorithms. I also consider the extent to which the distinctions revealed by learning speed are also revealed by geometric analyses such as RSA.

References

Cao, R., & Yamins, D. (2021). *Explanatory models in neuroscience: Part 2 – constraint-based intelligibility.* (arXiv:2104.01489)

Churchland, P. M. (1989). *A neurocomputational perspective: The nature of mind and the structure of science*. The MIT Press.

Churchland, P. S. (1986). *Neurophilosophy: Toward a unified science of the mind-brain*. The MIT Press.

Cormier, S. M., & Hagman, J. D. (1987). *Transfer of learning: Contemporary research and applications*. Academic Press.

De Lillo, C., Floreano, D., & Antinucci, F. (2001). Transitive choices by a simple, fully connected, backpropagation neural network: Implications for the comparative study of transitive inference. *Animal Cognition*, *4*, 61-68.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., … Olah, C. (2021). A mathematical framework for transformer circuits. *Transformer Circuits Thread*. (https://transformer-circuits.pub/2021/framework/index.html)

Geiger, A., Lu, H., Icard, T., & Potts, C. (2021). *Causal abstractions of neural networks.* (arXiv:2106.02997)

Horgan, T., & Tienson, J. (1996). *Connectionism and the philosophy of psychology*. MIT Press.

Hupkes, D., Dankers, V., Mul, M., & Bruni, E. (2020). *Compositionality decomposed: How do neural networks generalise?* (arXiv:1908.08351)

Kay, K., Biderman, N., Khajeh, R., Beiran, M., Cueva, C. J., Shohamy, D., … Abbott, L. (2023). Emergent neural dynamics and geometry for generalization in a transitive inference task. *bioRxiv*. doi: 10.1101/2022.10.10.511448

Lake, B. M., & Baroni, M. (2023). Human-like systematic generalization through a meta-learning neural network. *Nature*, *623*, 115–121.

Lillicrap, T. P., & Kording, K. P. (2019). *What does it mean to understand a neural network?* (arXiv:1907.06374)

Lippl, S., Kay, K., Jensen, G., Ferrera, V. P., & Abbott, L. (2023). A mathematical theory of relational generalization in transitive inference. *bioRxiv*. doi: 10.1101/2023.08.22.554287

Marr, D. (1983). *Vision: A computational investigation into the human representation and processing of visual information*. Henry Holt & Company.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*, 115–133.

Nanda, N., Chan, L., Lieberum, T., Smith, J., & Steinhardt, J. (2023). *Progress measures for grokking via mechanistic interpretability.* (arXiv:2301.05217)

Newell, A., & Simon, H. (1972). *Human problem solving*. Prentice-Hall.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., … Olah, C. (2022). *In-context learning and induction heads.* (arXiv:2209.11895)

Piccinini, G., & Maley, C. (2021). Computation in physical systems. In E. N. Zalta (Ed.), *The Stanford encyclopedia of philosophy.* Metaphysics Research Lab, Stanford University.

Ramsey, W. M. (2007). *Representation reconsidered*. Cambridge University Press.

Ruder, S. (2017). *An overview of multi-task learning in deep neural networks.* (arXiv:1706.05098)

Thrun, S., & Pratt, L. (2012). *Learning to learn*. Springer Science & Business Media.

Vafaeikia, P., Namdar, K., & Khalvati, F. (2020). *A brief review of deep multi-task learning and auxiliary task learning.* (arXiv:2007.01126)

von Neumann, J. (1958). *The computer and the brain*. New Haven: Yale University Press.

Wang, K., Variengien, A., Conmy, A., Shlegeris, B., & Steinhardt, J. (2022). *Interpretability in the wild: A circuit for indirect object identification in GPT-2 small.* (arXiv:2211.00593)

Zhou, C., Li, Q., Li, C., Yu, J., Liu, Y., Wang, G., … Sun, L. (2023). *A comprehensive survey on pretrained foundation models: A history from BERT to ChatGPT.* (arXiv:2302.09419)