DEEP NEURAL NETWORKS

Generalizing universal function approximators

At the heart of many challenges in scientific research lie complex equations for which no analytical solutions exist. A new neural network model called DeepONet can learn to approximate nonlinear functions as well as operators.

Irina Higgins

What do some of the hardest and most important problems in science have in common? From improving our ability to model turbulence and climate change, to developing nuclear fusion or superconductor technology, the solutions to all of these problems rely on being able to solve and evaluate complex ordinary or partial differential equations (ODEs or PDEs) and integrals. While this is not an issue for simple systems where the equations are well studied and analytical solutions exist, the majority of open problems do not share these properties. Hence, the equations are often approximated from noisy experimental data, and their evaluation around new data points is obtained from running complex numerical simulations on large supercomputers; both expensive options, computationally and financially. A system that could both estimate the equations from data, and evaluate them around new data points in a fast and reliable way could be revolutionary. In their recent work in *Nature Machine Intelligence*, Lu et al.[1] make a step towards developing such a system. They introduce Deep Operator Network (DeepONet), a neural network model that is capable of learning nonlinear operators that can, for example, evaluate integrals or solve differential equations directly from data in a way that generalizes to new inputs.

Deep neural networks have become one of the most powerful paradigms in machine learning. Their ability to recognize objects in images[2], use language[3] or play challenging games[4–6] can often surpass that of humans. The reason why deep learning works so well on these seemingly disconnected tasks is because these problems can be boiled down to one thing: complex function approximation. To recognize that an image depicts a cat or to decide what move to play next in chess, all that is necessary is to learn a complex function mapping from the space of real numbers representing the image or the board state, to the space of real numbers representing the index of the cat label, or the value of making the different available chess moves. While it has been known for decades that neural networks of arbitrary width[7] or depth[8] could approximate any continuous function mapping between real numbers — known as the universal approximation theorem — this knowledge did not translate into practice until the advances in computation hardware and optimization techniques allowed deep architectures to be efficiently trained[9].

Could this recipe for success be generalized beyond the existing function approximation abilities of neural networks to a broader set of mathematical operators, like being able to solve and evaluate integrals or differential equations? It appears that at least in theory, the answer is yes. A long-known theorem[10] shows that a neural network with a single hidden layer can accurately approximate any nonlinear continuous functional (a mapping from a space of functions into real numbers) or operator (a mapping from a space of functions into another space of functions) (Fig. 1a). In their paper, Lu et al. set about finding a practical implementation of this theoretical idea.

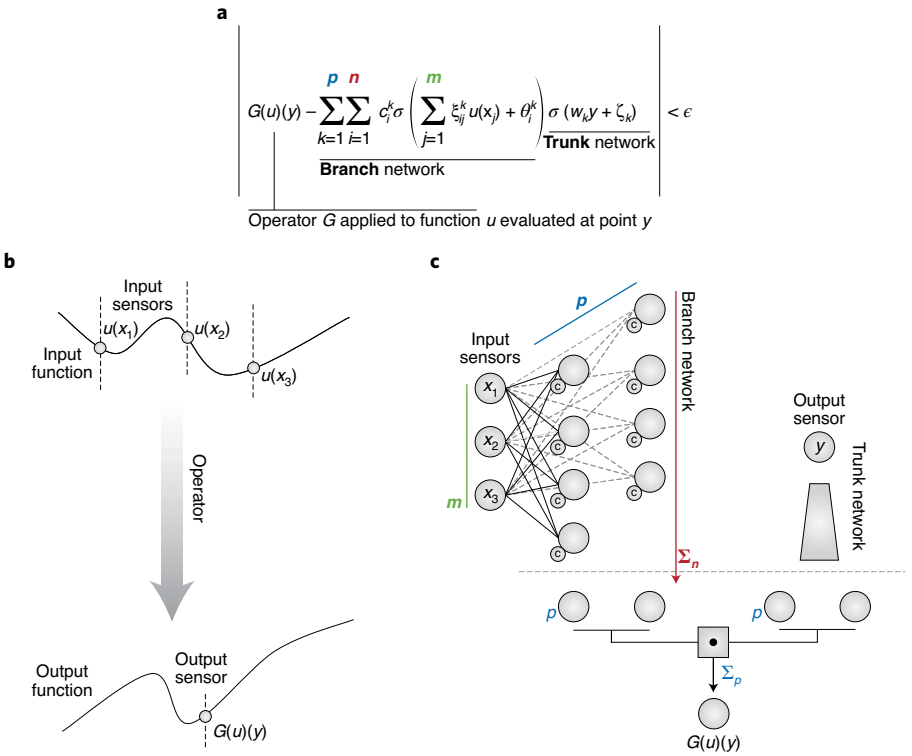

Fig. 1 | DeepONet translates the theoretically known ability of neural networks to approximate mathematical operators into a working implementation. a, Universal approximation theorem for operators[10] provides theoretical guarantees on the ability of neural networks to accurately approximate any nonlinear continuous operator — a mapping from a space of functions into another space of functions. **b**, DeepONet can map between continuous functions by using discrete 'sensors' to represent the input and output functions to the network. **c**, Lu et al.[1] translate the equation in **a** into the DeepONet architecture by designing and implementing the branch network to process the input function, and the trunk network to specify the constraints on the desired output when applying the learnt operator.

The first important problem the authors had to overcome when designing the model was to figure out how to present continuous functions to the network. The authors proposed using a discrete 'sensor' system, where the functions are represented by the value they take at a number of fixed sensor locations (Fig. 1b). For example, a function like $y = x^2$ can be represented by its value evaluated at three sensor locations: $x_1 = 2$, $x_2 = 5$ and $x_3 = 6$, resulting in an input vector $\mathbf{u}(x) = [4, 25, 36]$ that can be passed to the network. In their paper, the authors demonstrate that an optimal finite number of sensors will exist for different operators without compromising on the theoretical results of the approximation theorem[11]. An important difference from past work is that DeepONet can be flexible in terms of where these sensors are located; they do not have to be placed on a regular grid, as long as the same sensors are used for all the training and testing data. While the locations of the input sensors are fixed, it should be possible to evaluate the output function obtained by applying the learnt operator at any location. Hence, the network is given the location of the output sensor too. The input and output sensor information is processed by the branch and trunk networks respectively (Fig. 1c).

In their paper, the authors evaluate DeepONet on 16 different problems, including explicit operators like integration and implicit operators like nonlinear deterministic or stochastic ODEs and PDEs. For the latter, the trunk network was used to represent the boundary conditions, initial conditions or forcing terms. Through systematic tests, the authors were able to demonstrate that DeepONet improves its generalization performance exponentially as the size of the training dataset grows from 100 to 10,000 data points. Furthermore, once DeepONet learns a given operator, it has better generalization to new input functions compared to simpler feedforward or recurrent deep neural network baselines.

All the data presented in the paper was generated using numerical solvers, thus burdening DeepONet with the computational cost inherent to these solvers. However, once DeepONet is trained, it can be applied to new input functions, thus producing new results substantially faster than numerical solvers. Another benefit of DeepONet is that it can be applied to simulation data, experimental data or both, and the experimental data may span multiple orders of magnitude in spatio-temporal scales, thus allowing scientists to estimate dynamics better by pooling the existing data. While DeepONet is still only a first step towards building truly powerful and scalable universal operator approximators, it opens up exciting opportunities, like modelling the dynamics of complex systems where no analytical descriptions exist, for example social dynamics. Future work should get a better theoretical understanding of why DeepONets have small generalization errors, and how the size of the network may affect its ability to approximate operators. ❐

Irina Higgins ✉
Deepmind, London, UK.
✉*e-mail: irinah@google.com*

Published online: 18 March 2021
https://doi.org/10.1038/s42256-021-00318-x

References

1. Lu, L., Jin, P., Pang, G., Whang, W. & Karniadakis, G. *Nat. Mach. Intell.* https://doi.org/10.1038/s42256-021-00302-5 (2021).
2. Touvron, H., Vedaldi, A., Douze, M. & Jegou, H. In *Advances in Neural Information Processing Systems* Vol. 32 8252–8262 (NeurIPS, 2019).
3. Brown, T. et al. In *Advances in Neural Information Processing Systems* Vol. 33 1877–1901 (NeurIPS, 2020).
4. Mnih, V. et al. *Nature* **518**, 529–533 (2015).
5. Silver, D. et al. *Nature* **529**, 484–489 (2016).
6. Vinyals, O. et al. *Nature* **575**, 350–354 (2019).
7. Cybenko, G. *Math. Control Signals Syst.* **2**, 303–314 (1989).
8. Hornik, K., Stinchcombe, M. & White, H. *Neural Netw.* **2**, 359–366 (1989).
9. Krizhevsky, A., Sutskever, I. & Hinton, G. In *Advances in Neural Information Processing Systems* Vol. 25 1097–1105 (NeurIPS, 2012).
10. Chen, T. & Chen, H. *IEEE Trans. Neural Netw.* **6**, 911–917 (1995).