REVIEW ARTICLE
https://doi.org/10.1038/s42256-018-0006-z

nature machine intelligence

Designing neural networks through neuroevolution

Kenneth O. Stanley[1,2]*, Jeff Clune[1,3]*, Joel Lehman[1]* and Risto Miikkulainen[4,5]*

Much of recent machine learning has focused on deep learning, in which neural network weights are trained through variants of stochastic gradient descent. An alternative approach comes from the field of neuroevolution, which harnesses evolutionary algorithms to optimize neural networks, inspired by the fact that natural brains themselves are the products of an evolutionary process. Neuroevolution enables important capabilities that are typically unavailable to gradient-based approaches, including learning neural network building blocks (for example activation functions), hyperparameters, architectures and even the algorithms for learning themselves. Neuroevolution also differs from deep learning (and deep reinforcement learning) by maintaining a population of solutions during search, enabling extreme exploration and massive parallelization. Finally, because neuroevolution research has (until recently) developed largely in isolation from gradient-based neural network research, it has developed many unique and effective techniques that should be effective in other machine learning areas too. This Review looks at several key aspects of modern neuroevolution, including large-scale computing, the benefits of novelty and diversity, the power of indirect encoding, and the field's contributions to meta-learning and architecture search. Our hope is to inspire renewed interest in the field as it meets the potential of the increasing computation available today, to highlight how many of its ideas can provide an exciting resource for inspiration and hybridization to the deep learning, deep reinforcement learning and machine learning communities, and to explain how neuroevolution could prove to be a critical tool in the long-term pursuit of artificial general intelligence.

It is remarkable to consider that the 100-trillion-connection human brain is a product of evolution, a natural process without intelligent oversight or forethought. Although artificial neural networks have seen great progress in recent years, they remain distant shadows of the great cognitive masterpiece of natural evolution. How do we get from where we are to what is called artificial general intelligence (AGI), which roughly means artificial intelligence (AI) as smart as humans?

Current neural network research is largely focused on the fields of 'deep learning'[1,2] and 'deep reinforcement learning'[3,4]. In these fields, the dominant method for training neural networks is backpropagation[5], an efficient algorithm for calculating the loss function's gradient, which when combined with stochastic gradient descent (SGD) can modify each neural network weight to greedily reduce loss. This method has proven remarkably effective for supervised learning, and has also produced impressive reinforcement learning results.

An alternative approach, which draws inspiration from the biological process that produced the human brain, is to train neural networks with evolutionary algorithms. This field is called neuroevolution (although not all work combining neural networks and evolutionary computation was called neuroevolution historically, we use the term here broadly to encompass all such efforts) and enables important capabilities that are typically unavailable to gradient-based approaches. Such capabilities for neural networks include learning their building blocks (for example activation functions), hyperparameters (for example learning rates), architectures (for example the number of neurons per layer, how many layers there are, and which layers connect to which) and even the rules for learning themselves. Neuroevolution has other fundamental differences from traditional approaches, for example that it maintains a population of solutions during search, which grants it interestingly distinct benefits and drawbacks. Finally, because neuroevolution research has (until recently) developed largely in isolation from gradient-based neural network research, the range of unique, interesting and powerful techniques invented by the neuroevolution community can provide an exciting resource for inspiration and hybridization to the deep learning, deep reinforcement learning and machine learning communities. There has also been a surge of interest lately in hybridizing ideas from neuroevolution and mainstream machine learning, and to highlight this emerging direction we describe a few such efforts. We conclude with promising future research directions that advance and harness ideas from neuroevolution, including in combination with deep learning and deep reinforcement learning, that could catalyse progress towards the ambitious goal of creating AGI.

Our goal is to share with the broader machine intelligence community work from the neuroevolution community. Each of us has spent many years in this field, and this Review naturally reflects our experience and the interests that we have developed over this long period. Thus, rather than a dispassionate historical account of all work in this area, this Review is an exposition of our particular passions—the areas where we have concentrated because we think they are essential, and where we believe innovation will and should continue. Many great researchers have contributed to this field, and although we hope to have captured the key contributions of as many as possible, the entire field's results and ideas are greater than we can do justice to here.

Classic neuroevolution

A major inspiration for the investigation of neuroevolution is the evolution of brains in nature. By the 1980s, the notion of an artificial neural network was well established, and researchers began to ask whether these rough abstractions of brains themselves might be evolved artificially through evolutionary algorithms. Such algorithms, also well established by that time, borrow inspiration from evolution in nature to breed fitter (for example higher-scoring)

[1]Uber AI Labs, San Francisco, CA, USA. [2]University of Central Florida, Orlando, FL, USA. [3]University of Wyoming, Laramie, WY, USA. [4]Sentient Technologies, San Francisco, CA, USA. [5]The University of Texas at Austin, Austin, TX, USA. *e-mail: kstanley@uber.com; jeffclune@uber.com; joel.lehman@uber.com; risto.miikkulainen@sentient.ai

candidates in a population over many generations[6]. Through mutation and crossover, the population gradually evolves to increasing levels of fitness. Researchers saw in such algorithms an opportunity to optimize neural networks. To some extent, researchers were intrigued by the potential for an alternative to backpropagation, but they were also motivated by nature's own achievements through evolution, which remain unmatched by artificial systems. Although at first researchers focused on evolving only the weights of small, fixed-topology networks (as an alternative to backpropagation), interest quickly turned to more ambitious possibilities such as evolving the topology (the architecture) of neural networks[7,8], and even the dynamics of intra-life learning (that is, evolved rules for updating weights as an alternative to backpropagation-based reinforcement learning)[9,10]. Early algorithms for evolving network topology simply mutated weights stored in connection matrices[11], but the focus turned quickly to more sophisticated encodings for describing and manipulating graphs[12]. Indirect encodings also became popular, where the genome is a formula for generating a network rather than a direct (weight-by-weight) description of the network itself[13,14]. New representations gained popularity, such as the graphical programs in Cartesian genetic programming[15,16] and the implicit encoding of connectivity in analogue genetic encoding[17], which is inspired by genetic regulatory networks.

As the scope of attributes to evolve broadened, so did the need for algorithmic advancements to support the more ambitious ends. For example, the shift from evolving fixed topologies to increasingly complex ones created new challenges like crossing over structures (that is, combining the structures of two parent networks to create a parsimonious offspring network) with different topologies and protecting more complex structures from dying out of the population before allowing enough time for their weights to be optimized to reveal their true potential.

One approach that gained considerable traction by addressing these challenges is the NeuroEvolution of Augmenting Topologies (NEAT) algorithm[18]. It addressed the problem of crossing over variable topologies through historical marking (which tells the crossover operator which parts of two neural networks are similar and can thus be swapped) and prevented premature extinction of augmented structures through a mechanism called speciation. Solving these problems made evolving increasingly complex topologies more effective.

The early successes in the field often concerned evolving neural network controllers for robots, known as evolutionary robotics[19,20]. One prominent success was to produce the first running gait for the Sony Aibo robot[21]. Another was evolving the neural networks and morphologies of robots that were 3D-printed and could move around in the real world[22]. Notable accomplishments outside of evolutionary robotics include helping to discover through NEAT the most accurate measurement yet of the mass of the top quark, which was achieved at the Tevatron particle collider[23]. Neuroevolution also enabled some innovative video game concepts, such as evolving new content in real time while the game is played[24] or allowing the player to train non-player characters as part of the game through neuroevolution[25]. Neuroevolution has also been used to study open questions in evolutionary biology, such as the origins of the regularity, modularity and hierarchy found in biological networks like the neural networks in animal brains[26,27].

Although impressive, especially in their day, all of these successful applications involved tiny neural networks by modern standards, composed of hundreds or thousands of connections instead of the millions of connections commonly seen in modern deep neural network (DNN) research. A natural question is whether evolution is up to the task of evolving such large DNNs, which we address next.

The new era of neuroevolution at scale

An intriguing historical pattern is that many classic machine learning algorithms perform qualitatively different, and far better, once they are scaled to take advantage of the vast computing resources now available. The best-known example is that of deep learning. The algorithms and ideas for how to train neural networks, namely backpropagation[5] coupled with optimization tricks (for example momentum[28,29]) and important architectural motifs (for example convolution[30] or long short-term memory units (LSTMs)[31]), have been known for decades[1]. Before about 2012, these algorithms did not perform well for neural networks with more than a few layers[1,2]. However, once combined with faster, modern computers, including the speedups provided by graphics processing units (GPUs), and paired with large datasets, these algorithms produced great improvements in performance, which have generated most of the recent excitement about, and investment in, AI[1,2,32–34].

Research in recent years has similarly shown that neuroevolution algorithms also perform far better when scaled to take advantage of modern computing resources. As described next, scientists have found that neuroevolution is a competitive alternative to gradient-based methods for training deep neural networks for reinforcement learning problems. These results are important because they also foreshadow the potential for neuroevolution to make an impact across the spectrum of neural network optimization problems, but now at modern scale, including cases such as architecture search where differentiation (as used in most conventional deep learning) is not a clear solution.

Reinforcement learning involves AI agents learning by trial and error in an environment without direct supervision. Instead, they try different action sequences, receive infrequent rewards for those actions and must learn from this sparse feedback which future actions will maximize reward[4]. This type of learning is more challenging than supervised learning, in which the correct output for each input is given during training, and the main challenge is learning that mapping in a way that generalizes. Reinforcement learning, in contrast, requires exploring the environment to try to discover the optimal actions to take, including figuring out which actions lead to rewarding events, sometimes when the relevant actions and the rewards that they generate are separated by long time horizons, which is known as the credit-assignment problem[4]. Although algorithms have existed for decades to train reinforcement learning agents in problems with low-dimensional input spaces[4], there has recently been a surge of progress and interest in deep reinforcement learning, which involves DNNs that learn to sense and act in high-dimensional state spaces (for example raw visual streams that involve thousands or more pixel values per frame of video). The results that have had particularly large impact are that deep reinforcement learning algorithms can learn to play many different Atari video games[3] and learn how to make simulated robots walk[35–37].

In a surprise to many, Salimans et al.[38] showed that a recent form of evolution strategy (a classic evolutionary algorithm)[39] called a natural evolutionary strategy (NES)[40] performs competitively with the best deep reinforcement learning algorithms, including deep Q-networks (DQN)[3] and policy gradient methods (for example A3C)[41]. The NES in ref. [38] directly evolves the weights of a DNN of the same size as that used in the DQN and A3C Atari work (with over four million weight parameters that must be learned). The surprise was that an evolutionary algorithm could compete with gradient-based methods in such high-dimensional parameter spaces. However, because NES can be interpreted as a gradient-based method (it estimates a gradient in parameter space and takes a step in that direction), many did not conclude from this work that a pure (gradient-free) evolutionary algorithm can operate at DNN scale. That changed with the result that a simple genetic algorithm was also competitive with DQN and A3C (and evolution strategy) on Atari games and outperformed them on many games[42]. Moreover, on a subset of games, the genetic algorithm even outperformed later, more powerful versions of these algorithms[43,44]. The genetic algorithm is entirely gradient-free in that it contains

a population of agents (each a DNN parameter vector) that are independently mutated, and that reproduce more if their performance is better relative to others in the population. Both Salimans et al.[38] and Such et al.[42] additionally showed that these neuroevolution algorithms can be run more quickly than the original DQN and A3C algorithms because they are more parallelizable if one has sufficient computing resources[38,42]. In some cases, and compared to some algorithms (for example DQN, but not A3C[42]), evolutionary algorithms can be less sample efficient, but because they are extremely parallelizable, they can run far faster in real (wall clock) time (for example hours instead of days), albeit at the cost of requiring more computing resources[38,42].

Both ref. [38] and ref. [42] also show that directly evolving the weights of neural networks can solve the continuous control problem of robots learning to walk. Specifically, Salimans et al.[38] showed that NES performs well at enabling a humanoid robot controlled by a two-layer neural network to walk, and again did so faster in terms of wall clock time than competing reinforcement learning algorithms owing to evolution allowing better parallellization. Such et al.[42] showed that a GA can also perform well on this task, although with worse sample complexity. Mania et al.[45] showed that a simplified neuroevolution variant of evolution strategies, training a single-layer neural network (that is, learning a linear mapping from states to actions), produces state-of-the-art results on these simulated robot control tasks, outperforming complex, modern versions of policy gradient methods (for example trust region policy optimization[36], proximal policy optimization[37] and deep deterministic policy gradients[35]). That neuroevolution performs well for controlling robots is not surprising, given its long history of success in the field of evolutionary robotics[19–21,46,47].

Interest is also growing in ways to hybridize the gradient-based methods of deep learning with neuroevolution. Lehman et al.[48] recently introduced safe mutations through output gradients. It is based on the insight that evaluating a policy is often expensive (running an entire episode in a physics simulator or video game to see how it performs on a task, for example), but evaluating the outputs of a neural network is often inexpensive (one need only conduct 'forward passes' of the network in a few saved reference situations). In neuroevolution, random mutations are made to the policy (the mapping from inputs to actions, here represented by a DNN). Some of these mutations may have no effect on the behaviour (policy) of the network, and others might have major (and thus usually catastrophic) consequences on the policy (for example always outputting the same action). The insight behind safe mutations is that we can keep a reference library of states and actions, and (incurring only the slight cost of a forward and backward pass) use gradient information to scale the per-weight magnitude of mutations to make changes to the policy on the reference set that are neither too large nor too small. Lehman et al.[48] show that this approach improves the performance of evolutionary algorithms, including enabling the successful evolution of the weights of networks with over a hundred layers, which is unprecedented. Another hybridization that has been proposed runs variants of gradient-based reinforcement learning as the engine behind crossover and mutation operators within a neuroevolution algorithm[49]. Still another direction, which has been shown to perform well, combines the style of evolutionary algorithms (which search directly in the space of neural network parameters) with the style of policy gradient and Q-learning algorithms (which search in the space of actions and then change neural network parameters via backpropagation to make profitable actions more likely) by creating random parameter perturbations to drive consistent exploration (like evolutionary algorithms), but then reinforcing successful actions into weight parameters via backpropagation[50,51].

What is exciting about the successes described so far is that they were achieved with simple neuroevolution algorithms. However, the neuroevolution community has invented many sophisticated techniques that can greatly improve the performance of these simple algorithms. Many are based on the observation that evolution, both in its natural and computational instantiations, is an exciting engine of innovation[52], and these more modern techniques attempt to recreate that creativity algorithmically to search for better neural networks. As we discuss below, work has already begun that ports many of these ideas, including those that encourage diversity, novelty and intrinsic motivation[42,53,54], and these enhancements are improving performance. Other important ideas covered in this article include indirect encoding, a method for encoding very large structures[55], and the evolution of architectures[56,57] for networks trained by gradient descent.

Continuing to test the best ideas from the neuroevolution community at the scale of deep neural networks with modern amounts of computing power and data is likely to yield considerable additional advances. Moreover, combining such ideas with those from deep learning and deep reinforcement learning is a research area that should continue to deliver many breakthroughs. Each of the next sections describes what we consider to be the most exciting ideas from the neuroevolution community in the hope of encouraging researchers to experiment with them at DNN scales and to blend them with ideas from traditional machine learning.

Novelty and behavioural diversity

A hallmark of natural evolution is the amazing diversity of complex, functional organisms it has produced—from the intricate machinery of single-cell life to the massive collaborative union of cells that form animals of all sorts, including humans. In addition to being interesting in its own right, this massive parallel exploration of ways of life was probably critical for the evolution of human intelligence, because diversity is what makes innovation possible[58,59]. Thus a similar drive towards diversity is important when considering neuroevolution as a possible route to human-level AI. For these reasons, neuroevolution (and evolutionary computation as a whole) have long focused on diversity[60,61]. Indeed, by adapting a population of solutions, evolutionary algorithms are naturally suited to parallel exploration of diverse solutions.

Most initial work on diversity in neuroevolution focused on encouraging diversity in the 'genetic space'—that is, the space of parameters—with the goal of circumventing local optima. The idea is that if search has converged to a local optimum, then encouraging exploration away from that optimum may be enough to uncover a new promising gradient of improvement. Representative approaches include crowding[62], in which a new individual replaces the one most genetically similar to it, and explicit fitness-sharing[60], in which individuals are clustered by genetic distance and are punished by how many members are in their cluster.

Although sometimes effective, such parameter-space diversity often fails to produce a wide diversity of different behaviours[63], because there are infinite ways to set neural network weights that instantiate the same behaviour, owing to function-preserving rescaling of weights[64], permuting nodes[65] or redundant mappings (for example, many different weight settings can cause a robot to fall down immediately). In other words, while it is trivial to generate diverse (but similarly behaving) parameter vectors, escaping from local optima often requires exploration of diverse behaviours[63], as biological evolution does, and as is important in animal[66] and human problem solving[67].

As a result of this limitation to genetic diversity, more recent approaches directly reward a diversity of behaviours[63,68], and further research has led to related ideas such as directly evolving for desired qualities such as curiosity[54], evolvability[69] or generating surprise[70]. A representative approach[68] involves a multi-objective evolutionary algorithm[71,72] that rewards individuals both for increasing their fitness and for diverging from other individuals in experimenter-specified characterizations of behaviour in the domain. In this way, the

search can organically push different individuals in the population towards different trade-offs between exploring in relevant behavioural dimensions and optimizing performance.

One helpful step in developing new algorithms that explore the breadth of potential diversification techniques is to break out of the box wherein evolution is viewed mainly as an optimizer. Biological evolution is unlike optimization in the sense that it does not strive towards any particular organism. Indeed, one of its fundamental mechanisms is an accumulation of diverse novelty, bringing into question whether optimizing for a single optimal individual captures what enables evolution to discover rich and complex behaviour.

This alternate point of view recognizes that diversity is the premier product of evolution[63]. A popular neuroevolution algorithm that adopts this perspective, called 'novelty search'[63], rewards neural networks only for behaving in a way that is novel relative to individuals produced earlier in search. In other words, the search algorithm includes no pressure towards greater improvement according to a traditional fitness or performance measure. The idea is that as a whole, the population will spread over generations of evolution to span a wide range of behaviours. Although the gradient of divergence in the genetic space can be uninformative (because many different genomes can produce the same uninteresting behaviour), the gradient of behavioural novelty often contains useful domain information. In other words, to do something new often requires learning skills that respect the constraints of a domain; for example, learning to perform a new skateboard trick requires the balance and coordination that might be gained just by riding around. Indeed, in some reinforcement learning domains, searching only for behavioural novelty outperforms goal-directed search[53,63]. Although first applied to small networks, novelty search has recently been demonstrated to scale to high-dimensional reinforcement learning problems, where it improves performance[42,53], providing another example of how ideas from the neuroevolution community too can benefit from modern amounts of computation.

Building on the results from novelty search and the general perspective of evolution as diversity-driven, a new and expanding area of neuroevolution research explores 'quality diversity' algorithms[73]. In quality diversity, an algorithm is designed to illuminate the diversity of possible high-quality solutions to a problem—just as evolution has uncovered well-adapted organisms across countless environmental niches. For example, in this kind of search, if the quality objective is an creature's speed, discovering a fast cheetah would not preclude discovering a fast ant: both may locomote quickly relative to creatures with similar morphologies.

Examples of early quality diversity algorithms include novelty search with local competition (NSLC[74]) and the multi-dimensional archive of phenotypic elites (MAP-Elites[75]), which provide different ways to integrate a pressure to perform well within a diversifying search. NSLC modifies a multi-objective evolutionary algorithm, enabling optimizing a population for both diverse and locally optimal individuals (individuals that are well-performing relative to similar strategies). MAP-Elites is a simple but powerful algorithm that subdivides a space of possible behaviours into discrete niches, each containing a single champion that is the highest-performing agent of that type found so far. Competition is enforced only locally, but mutation to a parent from one niche can produce a new champion in another niche, enabling exaptation-like effects: that is, becoming high-performing in one niche may be a stepping stone to success in another. The product of such algorithms is often called a repertoire: that is, a collection of diverse yet effective options rather than a single optimal solution. In a result published in *Nature*, MAP-Elites was applied to discover such a diverse repertoire of high-performing walking gaits, so that after being damaged a robot could quickly recover by searching for the best of these champion gaits that worked despite the damage[47].

The space of quality diversity algorithms continues to expand[76–80] and is an exciting area of current research.

Although much progress has been made in diversity-driven neuroevolution algorithms, there remains a considerable qualitative gap between the complexity of what nature discovers and the current products of evolutionary algorithms. Such a gap hints that there are breakthroughs in this area yet to be made.

Indirect encoding

With about 100 trillion connections and 100 billion neurons[81], the human brain far exceeds the size of any modern neural network. Situated within its expanse is an intricate architecture of modules and patterns of connectivity that underpin human intelligence. A fascinating question is how this astronomical structure is encapsulated within our DNA-based genetic code, whose capacity is only about 30,000 genes (or 3 billion base pairs)[82]. Learning, of course, is a critical part of the story, but there is still a tremendous amount of information encoded by the genome regarding the overall architecture (how many neurons there are, their modular components, which modules are wired to which other modules and so on)[83]. The rules that govern how learning occurs is also part of the specification. The need to encode all these components requires regularity (that is, the reuse of structural motifs) and the compression that it enables, so that the genome can be reasonably compact.

Interestingly, regularity provides powerful computational advantages for neural structures as well. For example, the power of regular structure is familiar to anyone with experience in deep learning through the success of convolution[30]. Convolution is a particular regular pattern of connectivity, wherein the same feature detector is situated at many locations in the same layer. Convolution was designed by hand as a heuristic solution to the problem of capturing translation-invariant features at different levels of hierarchy[2]. This simple regularity has proven so powerful as to become nearly ubiquitous across the successful modern architectures of deep learning[2,84].

However, neuroevolution raises the prospect that the identification of powerful regularities need not fall ultimately to the hands of human designers. This prospect connects naturally also to the potential of compressed encoding to describe vast architectures composed of extensive regularities beyond convolution. For example, a larger palette of regularities could include various symmetries (bilateral or radial, for instance) as well as gradients along which filters vary according to a regular principle (such as becoming smaller towards the periphery). Ultimately it would be ideal if machine learning could discover such patterns, including convolution, on its own, without requiring (and being limited by) the cleverness of a designer or the reverse-engineering capabilities of a neuroscientist.

'Indirect encoding' in neuroevolution addresses this prospect by investigating the potential for artificial genetic encodings that can compactly capture regularities such as symmetries in structure. Motivated by the compression of DNA in nature, research in indirect encoding stretches back decades to experiments[85,86] in pattern formation. Later researchers explored evolvable encodings for a wide range of structures from blobs of artificial cells to robot morphologies to neural networks[55], including influential work by Gruau[7], Bongard and Pfeifer[87], and Hornby and Pollack[88].

A popular modern indirect encoding in neuroevolution is compositional pattern-producing networks (CPPNs[89]). CPPNs function similarly to neural networks, but their inspiration comes instead from developmental biology, where structure is situated and built within a geometric space. For example, early in the development of the embryo, chemical gradients help to define axes from head to tail, front to back, and left to right[90]. That way, structures such as arms and legs can be situated in their correct positions. Furthermore, within such structures are substructures, such as the fingers of the hand which themselves must be placed within the local coordinate

system of the hand. All of this configuration happens in biological systems through cells producing and reacting to diffusing chemicals called morphogens, which would be extremely computationally expensive to simulate.

CPPNs abstract this process into a simple network of function compositions that can be represented as a graph. At the input layer, the primary axes (for example x and y for a two-dimensional structure) are input into the network, serving as the base coordinate system. From there, a small set of activation functions that abstract common structural motifs within developing embryos is composed to yield more complex patterns For example, a Gaussian function elicits the equivalent of a symmetric chemical gradient, a sigmoid generates an asymmetric one, and a sine wave recalls segmentation. When such functions are composed with each other within a weighted network (like a special kind of neural network; Fig. 1a,b), they can yield surprisingly complex structures with very few nodes and connections by modern standards. For example, the CPPN in Fig. 1b produces the 'skull' image from Picbreeder (Fig. 1c; ref. [91]). Traditionally CPPNs are evolved with the NEAT[18] algorithm, which allows architectures of increasing complexity to evolve starting from a very simple initial form.

It is important to note that composing simple functions does not yield simple patterns. For example, while a sine wave encodes strict repetition, a sine composed with the square of a variable, $\sin(x^2)$, yields repetition with variation, a powerful and ubiquitous concept seen across biology. Thus a network of such functions quickly becomes complex, in part explaining how the composition of chemical gradients in a hierarchy in real organisms yields structures (including the brain) of such complexity.

CPPNs have been used in a wide range of applications that benefit from their tendency towards regular structure, from generating pictures[91] to creating three-dimensional objects[92], including the forms of soft robots[93]. They are also the method behind recent widely discussed results showing that some images (including those generated by CPPNs) can easily fool otherwise highly accurate DNNs[94]. Their ease of implementation and rich resultant structural spaces have furthermore inspired research on how to encourage creative innovation through evolution[80] and into the source of canalization (the tendency of indirect encoding in nature to yield robust, easily adaptable developmental pathways) in evolution[95]. They have even inspired fixes to some of the limitations of convolution in DNNs based on the idea of providing a coordinate space as inputs to DNNs[96].

Beyond these applications, perhaps the most important role of CPPNs is to generate the patterns of weights in neural networks themselves in an approach called HyperNEAT (hypercube-based NEAT[97,98]). The main idea is to generate the pattern of weights as a function of the geometry of the inputs and outputs of the domain. For example, if the input is a two-dimensional visual field, then the weights projecting from that field are generated as a function of the position of the source (input) neuron within that field. In short, if both the source field and target field are two-dimensional, then the weight of a connection between two neurons in them can be expressed as a function f (encoded as a CPPN) of the positions (x_1,y_1) and (x_2,y_2) of the source and target neurons, respectively, at either end of that weight: $w_{x_1,y_1\rightarrow x_2,y_2}=f(x_1,y_1,x_2,y_2)$ (Fig. 1d). To see how this formalism can enable regular connectivity patterns, consider that a simple symmetric function of x_2, for example $|x_2|$, which can be input as $f(x_1,y_1,|x_2|,y_2)$, causes the weights projecting from each source neuron to have patterns of weights to their targets symmetric about $x_2=0$. Interestingly, adding inputs x_1-x_2 and y_1-y_2 (or inputting only those) can induce a generalized convolutional connectivity pattern. There are many ways to extend the CPPN-based encoding to different neural architectures; for example, if there are multiple layers then their respective connectivity patterns can be generated by separate CPPN outputs.

Encoding connectivity as a function of geometry is in effect a means of conveying critical problem structure to the neuroevolution algorithm. For example, if there should be a correlation between weights of nearby neurons, then that can only be learned as a general pattern if the positions of the neurons are known. Convolution[30], perhaps the most ubiquitous heuristic in modern deep neural networks, is itself a repeated pattern across the geometry of the neural structure, and in fact recent variants of HyperNEAT-based neuroevolution that combine neuroevolution and indirect encoding with SGD have indeed discovered convolutional patterns on their own (without explicitly inputting a convolutional coordinate frame), in effect reinventing the idea[99]. Ha et al.[100] recently also introduced hypernetworks, which adapt the HyperNEAT indirect encoding to train weights entirely through SGD. In ref. [100], hypernetworks help to enhance the performance of LSTMs in language modelling and other tasks, providing an example of how ideas from neuroevolution are being fruitfully hybridized with ideas from the traditional machine learning community.

What makes HyperNEAT interesting is that convolution is not the only conceivable regular pattern of connectivity that could be important. HyperNEAT-evolved neural networks can in principle discover any such pattern and thereby exploit different kinds of regularities not accessible to conventional neural network learning algorithms. For example, the positions of sensor and motor neurons within a quadruped body can be exploited to efficiently evolve regular gait patterns[46], which require regular connectivity patterns unrelated to convolution. Another example is that HyperNEAT can create repeated architectural motifs, such as a repeated modular design[27].

Another great benefit of HyperNEAT and indirect encoding in general is that they enable very large neural networks to be evolved through compact encoding (which connects back to the inspiration from DNA). For example, HyperNEAT can generate functional neural networks with millions of connections that are encoded by CPPNs with only dozens of connections[98]. This potential for extreme compression inspired researchers who subsequently developed other indirect encodings that generate patterns over network geometry, such as the weight compression-based indirect encoding of Steenkiste et al.[101] and Koutnik et al.[102], which is among the first systems to approach reinforcement learning directly from pixels to actions. Interestingly, although the work of DeepMind[3] had a strong impact on the field of reinforcement learning for learning to play Atari directly from pixels, HyperNEAT was the first system for which direct pixel-to-action Atari results were reported[103]. Research also continues on other indirect encodings for neural networks, such as Cartesian genetic programming[15,16,104].

Today's increasingly powerful computation presents great opportunities for indirect encoding. Not only can increasingly large neural networks be held in memory, but the decoding step wherein the CPPN generates the weights of the neural network (the analogue of embryogeny in nature) can also be accelerated and parallelized. Newfound computational capacity also opens up more sophisticated uses of indirect encoding that nudge neuroevolution even closer to natural brain-like evolution, such as using the CPPN to generate patterns of coefficients of plasticity instead of static weights, as in 'adaptive HyperNEAT'[105]. There are also variants of HyperNEAT that can encode and thus learn neural network architectures and their weights at arbitrary resolutions, such as ES-HyperNEAT[106].

Evolving neural networks that can learn over their lifetime opens up broad new opportunities for powerful meta-learning (discussed next), moving the field closer to the kind automatic design of entire learning systems that could fundamentally alter the landscape of deep learning.

Meta-learning and architecture search

While meta-learning[107], or 'learning how to learn', is currently experiencing a surge of interest within deep learning[108–111], it has long

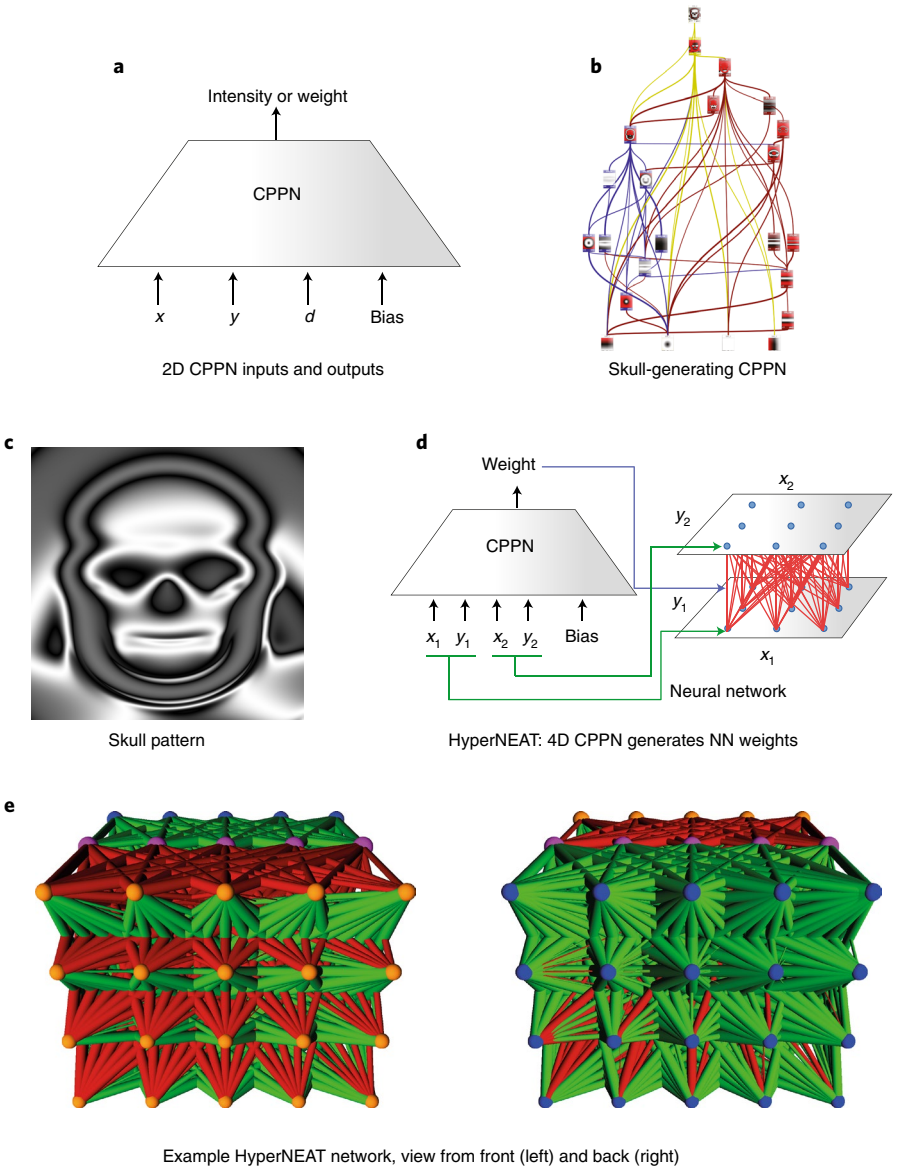

Fig. 1 | Compositional pattern-producing networks and HyperNEAT. CPPNs are networks of heterogeneous activation functions that produce a geometric pattern. **a**, The network shown here is a two-dimensional (2D) CPPN (because it takes *x* and *y*) that also takes the input *d*, which is the distance of (*x*,*y*) from the image centre. When queried at many points, its output is interpreted as an image or pattern in space. **b**,**c**, The CPPN shown in **b** is the actual architecture that produces the skull pattern shown in **c**, which was interactively evolved through the Picbreeder online picture-breeding service[91]. The colours in **b** distinguish components of the evolved network responsible for key parts of the image in **c** based on an analysis of function, and the small images within the nodes are the activation patterns computed by those specific nodes in (*x*,*y*), which are ultimately combined by the network to produce the final image in **c**. **d**, In HyperNEAT, CPPNs leverage their pattern-generating capabilities to output a weight (blue arrow) for every source and target node location (green arrows) in a neural network. **e**, An example HyperNEAT neural network evolved to make a quadruped robot run rapidly[46], viewed with either the input neurons (orange spheres) or output neurons (blue spheres) in front. Pink spheres are hidden neurons. Green (red) connections are excitatory (inhibitory), and their thickness indicates connection strength. Note the complex, regular, geometric weight patterns (for example, outgoing connections from inputs are inhibitory to upper hidden nodes and excitatory towards lower hidden nodes) and regularity with variation (such as the diffusion of inhibitory connections into output nodes, which varies across both the *x* and *y* axes).

attracted the attention of researchers in neuroevolution. Natural evolution, after all, is intrinsically a powerful meta-learning algorithm. Evolution can be viewed as an outer loop search algorithm that produced organisms (including humans) with extraordinarily sophisticated learning capabilities of their own—that is, they can be seen as running their own inner loop search. Thus, it is natural that researchers inspired by evolution in nature have explored the potential for a similar nested loop of meta-learning within artificial systems. Perhaps that way, brain-like structures with powerful learning capabilities could arise on their own.

Much of the meta-learning work within neuroevolution has focused on synaptic plasticity, which is a key mechanism behind neural learning in nature[10]. Early experiments[112,113] explored the potential to evolve local learning rules that dictate how the weights of connections should change in response to the activation levels of their source and target nodes. The most famous such rule is the Hebbian rule[114], which, roughly, implements synaptic strengthening when the source and target neurons fire together. Although the Hebbian rule can be expressed in a variety of forms, the simple expression $\Delta w_{i \to j} = \eta x_i x_j$ describing the change in the

weight from neuron i to neuron j as a function of their activations (x_i and x_j) is sufficient to show where there is room for evolution to decide the dynamics of the network: the coefficient η for each connection can be evolved such that the degree of plasticity of each connection is optimized.

Various local learning rules are possible. In effect, any conceivable function can determine the dynamics of a synapse. In fact, it is even possible for an indirect encoding to generate a pattern of learning rules as a function of the geometry of network connectivity, as mentioned above with adaptive HyperNEAT[105]. Furthermore, the plasticity of connections themselves can be made to vary over the lifetime of the agent through the mechanism of neuromodulation: that is, a special neuromodulatory signal can increase or decrease plasticity in individual connections. This capability is powerful because it assists a kind of reward-mediated learning (similar to reinforcement learning), for example by locking in weights (by turning off plasticity) when they yield high rewards, or increasing plasticity when expected reward does not materialize. All these mechanisms can unfold in concert with recurrence inside the network, providing evolution with a rich playground for designing complex plastic systems that can learn. Soltoggio et al.[115] pioneered evolving neuromodulatory plasticity, and others have followed up, including showing the benefits of combining plasticity with indirect encoding[116,117] (see ref. [10] for a comprehensive review of this burgeoning research area and its achievements so far).

One exciting opportunity that neuromodulation enables is the mitigation of 'catastrophic forgetting', which is a grand challenge in machine learning that must be solved to create AGI. Natural animals are able to learn a variety of different tasks across their lifetimes and remember how to perform an already learned task even if they have not done it for a long while[118]. In sharp contrast, when artificial neural networks learn new skills, they do so by erasing what they have learned about previous skills, meaning they forget catastrophically[119]. Neuromodulation offers the tantalizing prospect of turning plasticity on only in the subset of neural weights relevant for the task currently being performed, meaning that knowledge stored in other weights about different tasks is left untouched. Ellefsen et al.[120] showed that combining neuromodulation with techniques that promote functional modularity in neural networks[26] (meaning that different tasks are performed in different modules within a neural network) mitigates catastrophic forgetting.

However, the effect was limited because the neuromodulation in ref. [120] was encoded separately for each connection, making it a challenging optimization problem for evolution to learn to shut off all of the connections in each task-specific module when that task is being performed. Velez et al.[121] introduced the idea of diffusion-based neuromodulation, which enables evolution to instead increase or decrease plasticity in different geometric regions of the neural network. The addition of this ability to easily upregulate or downregulate plasticity in entire geometric regions enabled evolution to create neural networks that entirely eliminate catastrophic forgetting, albeit in simple neural networks solving simple problems. Testing this technique at the scale of DNNs is an exciting area of future work, and yet another example where neuroevolution could benefit from modern computation. Additionally, although promising work has appeared recently in deep learning to combat catastrophic forgetting[122,123], diffusion-based neuromodulation is an entirely different approach to the problem that can be divorced from neuroevolution and coupled instead to SGD, providing another potential example of porting ideas from the neuroevolution community to deep learning.

Even with indirect encoding, much of the work so far on evolving learning mechanisms through synaptic plasticity has focused on relatively small neural networks solving relatively simple problems (such as using plastic connections to remember where food is within a simple maze[115]), but the recent results showing neuroevolution algorithms optimizing millions of weights[38,42,53] hint that much larger plastic systems could now be possible to evolve. The full capabilities of such complex dynamic systems are yet to be explored, setting up an exciting near-future opportunity. Aside from plasticity, another aspect of learning to learn is discovering the right architecture for learning. Much work in neuroevolution has also focused on this challenge of architecture search. Network architectures have become so complex that it is difficult to design them by hand—and such complexity matters. While NEAT represents an early advance in evolving network architecture (along with weights) at small scales, recent work has focused on evolving deep neural networks[56,57,124]. They evolve the network architectures and optimize their weights with gradient descent, in effect optimizing large-scale networks for the purpose of gradient descent. Many methods are inspired by NEAT's ability to increase complexity over generations, but do so by adding layers instead of individual neurons. Such optimization has already led to architectures that improve the state of the art in several deep learning benchmarks, including those in vision, language and multitask learning.

Many of the greatest achievements in deep learning have been in the visual domain—in image processing tasks such as classification, segmentation or object detection. Neural networks have come to dominate computer vision and have led to substantial performance improvements in these different domains[1]. Often innovations within a computer vision domain come from the discovery of different, better architectures, and each type of computer vision problem tends to require its own specialized architecture[125–127]. The result has been a proliferation of neural network architectures for image processing. These architectures are complex and varied enough to raise the question of whether better architectures could be discovered automatically. Recent results are showing that they can be.

A particularly successful approach[128] took inspiration from NEAT and evolved DNNs by starting small and adding complexity through mutations that added entire layers of neurons, achieving impressive performance on the popular CIFAR image classification dataset. A variant of the approach[57] further improved performance by evolving small neural network modules that are repeatedly used in a larger hand-coded blueprint. The blueprint[129] was inspired by the idea of repeatedly stacking the same layer modules to make a DNN, an idea that has proved successful in the high-performing Inception[127], DenseNet[126], and ResNet[130] architectures. For efficiency reasons, these modules were evolved on a computationally cheaper task that can be solved with a smaller blueprint and were then transferred (with no further evolution) to build a larger network to solve the target, more computationally expensive tasks of image classification on the CIFAR and ImageNet datasets. Despite massive efforts by armies of researchers over years to hand-design architectures for these famous computer vision benchmarks, this method and its evolved architectures produced state of the art at its time of publication on CIFAR (it has since been slightly surpassed[131]) and currently holds the state of the art on ImageNet[57].

Language processing is another area where deep learning has had a large impact. The architectural needs, however, differ from those of vision. Whereas convolutional filters are the key architectural motif in vision, in language the most common motifs are gated recurrent networks, such as the LSTM[31]. A node in such networks is more complex than in other neural networks. Instead of a weighted sum followed by a nonlinearity, it typically includes a memory cell and connections for write, read and forget gates. Such a structure makes it easier to retain activation values indefinitely (improving gradient flow through the network), and makes it possible to perform well on various sequence processing tasks. Even though these ideas originated in the 1990s, it was not until the computation was available to scale them up to tens of millions of parameters and train them at length on large datasets that they started to work well enough to make a difference in real-world applications such as speech recognition, language understanding and language translation[132].

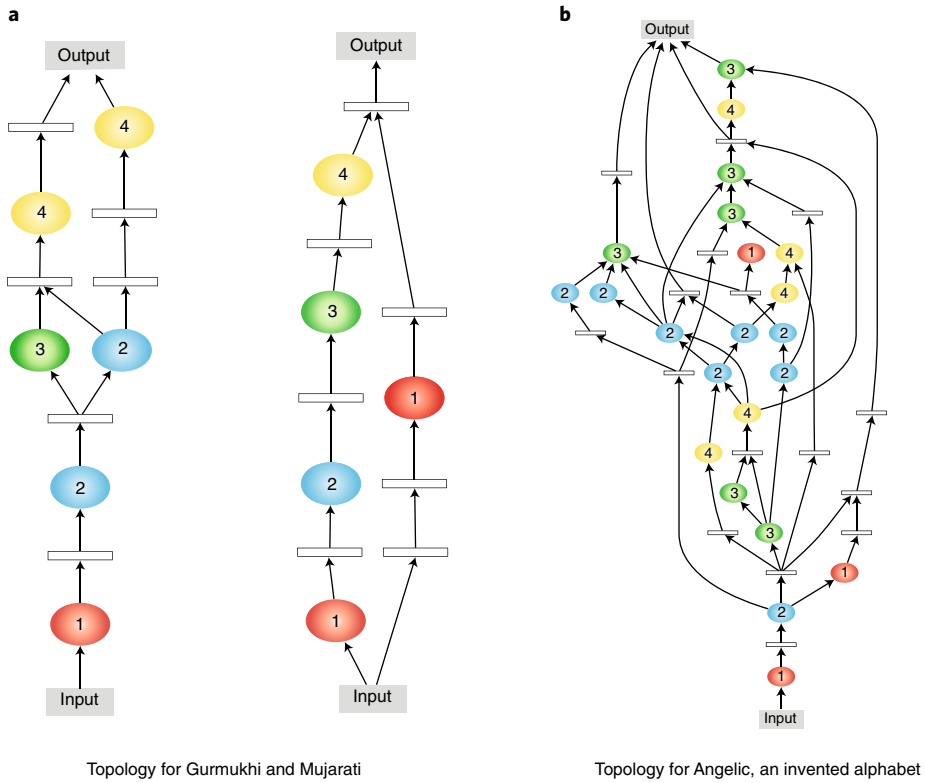

Fig. 2 | Sample evolved topologies of modules for the Omniglot multitask learning benchmark. Neuroevolution discovers both a set of common building blocks (the modules, not shown but identified by number) and a differentiation between tasks, making it possible to perform better in each task even with limited training data[139]. **a**,**b**, The topologies for each task that combine these modules represent a range from simple (**a**) to complex (**b**); similar alphabets (Gurmukhi, left, and Mujarati, right) have similar topologies, and the structure is consistently found in different independent evolutionary runs. Related neuroevolution techniques similarly improve state of the art in various vision and language machine learning benchmarks. Such useful task-specific architectural specialization would be difficult and expensive to discover by hand, demonstrating the power of evolution in designing complex systems.

Interestingly, the structure of the LSTM node has remained relatively constant since its inception 25 years ago. The variations that have been proposed do not significantly improve on the standard LSTM[133,134]. In the past few years, however, it has become possible to search automatically for better gated recurrent network designs, using, for example, reinforcement learning and evolution[124,135].

A fruitful approach is to encode the structure of the gated recurrent node as a program that is evolved through genetic programming—representing the genes that describe the network in the form of a program[56]. Given a search space with multiple memory cells, reusable inputs, different activation functions and the ability to construct multiple linear and nonlinear paths through the node, evolution came up with complex structures that are several times larger than the standard LSTM. These nodes can then be put together into layered networks and trained in, for example, the language modelling task, which is a machine learning benchmark of predicting the next word in a large corpus of text[136]. Even when the total number of adjustable parameters is the same (20 million), the evolved nodes perform 15% better than standard LSTM nodes[56]. Remarkably, when the same node design is used in another task such as music prediction, it does not improve over standard LSTMs—but if the node structure is evolved again in music prediction itself, performance improves by 12%[56]. In other words, neuroevolution discovers complexity that is customized to, and improves performance on, each task.

Another promising application of architecture evolution is multitask learning. It has long been known that training a neural network on multiple tasks at once can make learning each task easier[137]. The requirements and data from each task help to shape internal representations that are more general and generalize better to new inputs. More generally, recent work has shown, theoretically and in practice, that combining gradients from each task improves learning[138].

But how should the requirements of the different tasks be best combined? The architecture of the multitask learning network can make a large difference, and therefore it is a good opportunity for architecture search. The most straightforward approach would be to evolve a single shared architecture with separate decoders (output layers) for each task. An alternate, promising method is CMTR (coevolution of modules and task routing[139]), which is based on three principles: (1) evolve custom network topologies for each task, (2) construct these topologies from modules whose training is shared across tasks and (3) evolve the structure of these modules. This method allows evolution to discover general modules and task-specific topologies.

The Omniglot task—recognizing characters in 50 different alphabets with few examples per class—has recently emerged as a standard benchmark in multitask learning. The CMTR approach improved the state of the art by 32% in this benchmark. Demonstrating the benefit of multitask training, on average the performance in each alphabet was 14% better than training a network with a similar size on each alphabet alone. Modules with very different topologies were discovered and used (Fig. 2). The topologies varied a lot between alphabets, but remarkably, topologies were consistent across multiple runs, and similar topologies were discovered for visually similar alphabets. These result suggest that the approach discovers useful common representations and specialized ways of using them in each task.

There are several alternate approaches for architecture optimization in neural networks besides evolution, including optimizing some part of the network design using gradient descent, reinforcement learning or Bayesian parameter optimization[140]. However, evolution is a promising method for exploring neural network structures most broadly because it does not require explicit gradients. Additionally, it can straightforwardly be augmented with novelty search, quality diversity, indirect encoding and other ideas discussed above, which could further improve results above those just discussed, which were all accomplished with simple evolutionary algorithms.

The prospect of combining the ability to evolve both neuromodulated plasticity and network architectures yields a unique new research opportunity that begins to resemble the evolution of brains. What this kind of technology can produce at scale, and when combined with gradient descent, remains largely unexplored.

Looking forward

Although neuroevolution has long existed as an independent research area, a trend is emerging wherein the line between neuroevolution and deep learning, both of which concern neural networks, gradually blurs. Ideas from neuroevolution infuse deep learning, both through hybridization of evolution and gradient descent, and through conceptual insights transferring from one paradigm to the other. This trend is likely to accelerate especially as computational resources expand, opening up evolutionary experiments that once focused on small models to large-scale discovery. Just as evolution and neural networks proved a potent combination in nature, so does neuroevolution offer an intriguing path to recapitulating some of the fruits of evolving brains.

The trend towards hybridization between neuroevolution and gradient-based methods encompasses many of the ideas in meta-learning, such as architecture search for gradient descent, but also manifests other unique and original forms[49,141–143]. One example is where the hyperparameters of gradient-descent based learning are evolved online in a population of different learners[144,145]. Another example is PathNet[141], an evolutionary algorithm that evolves pathways online that are trained by gradient descent, enabling automatic discovery of shared features for transfer learning; evolved policy gradients[142] evolves the loss function for a policy gradients algorithm, enabling the discovery of internal reward signals that accelerate learning on new tasks (just as evolution endows us with innate desires, such as enjoying eating, and intrinsic motivation, such as curiosity, that improve our ability to learn and survive). Other methods instantiate evolution with gradient-descent-based mutation and recombination operators through policy gradients[49] or in GAN training[143]. Gradient descent can also boost the performance of evolution, such as in the gradient-based 'safe mutation' operators introduced in ref. [48], which avoid overly disruptive mutations by estimating the sensitivity of individual weights to perturbation through a gradient computation.

The other side of the convergence between neuroevolution and deep learning is the importation of conceptual approaches from neuroevolution into gradient-based implementations[51,100,110,146] without using any evolution. For example, plastic neural networks, that is, neural networks that encode their own online learning rules such that weights change (without gradient descent) in response to experience, have long been studied within artificial life and neuroevolution[9,113,115], but only recently have similar plastic networks been reformulated such that they can be learned by gradient descent (which also enables their impact to be explored on a much larger scale than before)[110,147]; as mentioned above, a similar translation has occurred with HyperNEAT[100] and CPPNs[96,99,148]. What drives this trend is probably that evolution is an open playground for testing and proving ideas, as it imposes few limitations on imaginable models (for example, it does not require that a model be differentiable), but ultimately, if it is possible to translate such ideas into gradient descent, then the effort may result in greater scalability and efficiency.

Neuroevolutionary ideas have also inspired progress in other areas of deep reinforcement learning. For example, the 'diversity is all you need' approach[146] mirrors similar insights from novelty search[63]. In other cases, older ideas from evolutionary computation are being reinvented independently in deep learning, offering an opportunity for diversity, cross-pollination and deeper understanding. An example is the recent, exciting self-play results[149,150], which resemble investigations in the subfield of coevolution[151–154]. We believe that such previous evolutionary work will continue to be a source of synergistic inspiration for future deep learning research. Additionally, the ideas developed in neuroevolution can be, and in some cases already have been[155], ported to improve subfields that optimize other sorts of networks, such as genetic regulatory networks.

A final critical opportunity for neuroevolution is to lead the effort to construct 'open-ended' algorithms. The main challenge in open-endedness is to create algorithms that produce interesting and increasingly complex discoveries indefinitely. The inspiration for open-endedness is evolution in nature, where all of life on Earth was discovered in a single run that has continued to produce new forms for over a billion years. If we can learn how to program algorithms that are similarly perpetually interesting, we could capture a profoundly powerful aspect of the creativity of nature, and direct it for our own purposes. While a straightforward application of open-endedness is to generate new artefacts such as buildings, clothing and architectures without bound, a more intriguing possibility is that because of its complexity, AGI itself is only possible to discover through an open-ended process that generates more and more complex brain-like structures indefinitely. Furthermore, open-endedness may require more than only neural networks to evolve—brains and bodies evolve together in nature, and so can morphologies evolve along with neural networks in artificial systems[156], providing a form of embodiment[13,157,158]. In the long run, open-endedness could be the fuel for generating the architectures and learning algorithms that ultimately reach human-level intelligence. The race is on now in neuroevolution research to develop the mechanisms of open-endedness that can innovate without boundaries, which we consider a grand challenge of scientific enquiry[159].

Many of the ideas discussed herein were invented at a time in which the available computation made it difficult to try them in anything but tiny neural networks. However, as with computer vision a few years ago, the computation is now becoming available to see these original ideas, and modern improvements on them, finally take off and achieve the grand ambitions we have long had for them. Just as evolution did in the natural evolution of intelligence, we expect that neuroevolution will play an important role in our collective quest to create human-level AI and algorithms that endlessly innovate.

Received: 6 September 2018; Accepted: 12 November 2018;
Published online: 7 January 2019

References

1. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* (MIT Press, Cambridge, 2016).
2. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
3. Mnih, V. et al. Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015).
4. Sutton, R. S. & Barto, A. G. *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, 2018).
5. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986).
6. De Jong, K. A. *Evolutionary Computation: A Unified Perspective* (MIT Press, Cambridge, 2002).

7. Gruau, F. Automatic definition of modular neural networks. *Adapt. Behav.* **3**, 151–183 (1994).
8. Yao, X. A review of evolutionary artificial neural networks. *Int. J. Intell. Syst.* **8**, 539–567 (1993).
9. Floreano, D., Dürr, P. & Mattiussi, C. Neuroevolution: from architectures to learning. *Evol. Intell.* **1**, 47–62 (2008).
10. Soltoggio, A., Stanley, K. O. & Risi, S. Born to learn: the inspiration, progress, and future of evolved plastic artificial neural networks. *Neural Netw.* **108**, 48–67 (2018).
11. Dasgupta, D. & McGregor, D. Designing application-specific neural networks using the structured genetic algorithm. In *Proc. COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks* 87–96 (IEEE, 1992).
12. Pujol, J. C. F. & Poli, R. Evolving the topology and the weights of neural networks using a dual representation. *Appl. Intell. J.* **8**, 73–84 (1998).
13. Bongard, J. C. & Pfeifer, R. in *Morpho-functional Machines: The New Species* (eds Hara, F. & Pfeifer, R.) 237–258 (Springer, Tokyo, 2003).
14. Gruau, F. Genetic synthesis of modular neural networks. In *Proc. 5th International Conference on Genetic Algorithms* (ed. Forrest, S.) 318–325 (Morgan Kaufmann, San Francisco, 1993).
15. Khan, M. M., Ahmad, A. M., Khan, G. M. & Miller, J. F. Fast learning neural networks using cartesian genetic programming. *Neurocomputing* **121**, 274–289 (2013).
16. Turner, A. J. & Miller, J. F. Neuroevolution: evolving heterogeneous artificial neural networks. *Evol. Intell.* **7**, 135–154 (2014).
17. Mattiussi, C. & Floreano, D. Analog genetic encoding for the evolution of circuits and networks. *IEEE Trans. Evol. Comput.* **11**, 596–607 (2006).
18. Stanley, K. O. & Miikkulainen, R. Evolving neural networks through augmenting topologies. *Evol. Comput.* **10**, 99–127 (2002).
19. Moriarty, D. E. & Miikkulainen, R. Evolving obstacle avoidance behavior in a robot arm. In *From Animals to Animats 4: Proc. 4th International Conference on Simulation of Adaptive Behavior* (eds Maes, P. et al.) 468–475 (MIT Press, Cambridge, 1996).
20. Nolfi, S. & Floreano, D. *Evolutionary Robotics* (MIT Press, Cambridge, 2000).
21. Hornby, G *et al.* Evolving robust gaits with AIBO. In *Proc. IEEE Conference on Robotics and Automation* 3040–3045 (IEEE, 2000).
22. Lipson, H. & Pollack, J. B. Automatic design and manufacture of robotic lifeforms. *Nature* **406**, 974–978 (2000).
23. Aaltonen, T. et al. Measurement of the top quark mass with dilepton events selected using neuroevolution at CDF. *Phys. Rev. Lett.* **102**, 2001 (2009).
24. Togelius, J., Yannakakis, G. N., Stanley, K. O. & Browne, C. Search-based procedural content generation: a taxonomy and survey. *IEEE Trans. Comput. Intell. AI Games* **3**, 172–186 (2011).
25. Stanley, K. O., Bryant, B. D. & Miikkulainen, R. Real-time neuroevolution in the NERO video game. *IEEE Trans. Evol. Comput.* **9**, 653–668 (2005).
26. Clune, J., Mouret, J.-B. & Lipson, H. The evolutionary origins of modularity. *Proc. R. Soc. B* **280**, 20122863 (2013).
27. Huizinga, J., Mouret, J.-B. & Clune, J. Evolving neural networks that are both modular and regular: Hyperneat plus the connection cost technique. In *Proc. Genetic and Evolutionary Computation Conference (GECCO)* 697–704 (ACM, 2014).
28. Polyak, B. T. Some methods of speeding up the convergence of iteration methods. *USSR Comput. Math. Math. Phys.* **4**, 1–17 (1964).
29. Qian, N. On the momentum term in gradient descent learning algorithms. *Neural Netw.* **12**, 145–151 (1999).
30. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998).
31. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997).
32. Dahl, G., Yu, D., Deng, L. & Acero, A. Context-dependent pre-trained deep neural networks for large vocabulary speech recognition. *IEEE Trans. Audio Speech Lang. Process.* **20**, 30–42 (2012).
33. Hinton, G. et al. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *IEEE Signal Process. Mag.* **29**, 82–97 (2012).
34. Krizhevsky, A., Sutskever, I. & Hinton, G. E. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25 (NIPS 2012)* (eds. Pereira, F. et al.) 1097–1105 (NIPS, 2012).
35. Lillicrap, T. P. et al. Continuous control with deep reinforcement learning. Preprint at https://arxiv.org/abs/1509.02971 (2016).
36. Schulman, J., Levine, S., Abbeel, P., Jordan, M. & Moritz, P. Trust region policy optimization. *J. Mach. Learn. Res.* **37**, 1889–1897 (2015).
37. Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. Proximal policy optimization algorithms. Preprint at https://arxiv.org/abs/1707.06347 (2017).
38. Salimans, T., Ho, J., Chen, X. & Sutskever, I. Evolution strategies as a scalable alternative to reinforcement learning. Preprint at https://arxiv.org/abs/1703.03864 (2017).
39. Rechenberg, I. in *Simulationsmethoden in der Medizin und Biologie* 83–114 (Springer, Hannover, 1978).
40. Wierstra, D. et al. Natural evolution strategies. *J. Mach. Learn. Res.* **15**, 949–980 (2014).
41. Mnih, V. et al. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning* 1928–1937 (PMLR, 2016).
42. Such, F. P. et al. Deep neuroevolution: genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. Preprint at https://arxiv.org/abs/1712.06567 (2017).
43. Hessel, M. et al. Rainbow: combining improvements in deep reinforcement learning. In *Proc. 2018 AAAI Conference on Artificial Intelligence* (AAAI, 2017).
44. Horgan, D. et al. Distributed prioritized experience replay. In *Proc. 2018 International Conference on Learning Representations* (OpenReview, 2018).
45. Mania, H., Guy, A. & Recht, B. Simple random search provides a competitive approach to reinforcement learning. Preprint at https://arxiv.org/abs/1803.07055 (2018).
46. Clune, J., Stanley, K. O., Pennock, R. T. & Ofria, C. On the performance of indirect encoding across the continuum of regularity. *IEEE Trans. Evol. Comput.* **15**, 346–367 (2011).
47. Cully, A., Clune, J., Tarapore, D. & Mouret, J.-B. Robots that can adapt like animals. *Nature* **521**, 503–507 (2015).
48. Lehman, J., Chen, J., Clune, J. & Stanley, K. O. Safe mutations for deep and recurrent neural networks through output gradients. In *Proc. Genetic and Evolutionary Computation Conference (GECCO)* 117–124 (ACM, 2018).
49. Gangwani, T. & Peng, J. Genetic policy optimization. In *Proc. 2018 International Conference on Learning Representations* (OpenReview, 2018).
50. Fortunato, M. et al. Noisy networks for exploration. In *Proc. 2018 International Conference on Learning Representations* (OpenReview, 2018).
51. Plappert, M. et al. Parameter space noise for exploration. In *Proc. 2018 International Conference on Learning Representations* (OpenReview, 2018).
52. Lehman, J. et al. The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. Preprint at https://arxiv.org/abs/1803.03453 (2018).
53. Conti, E. et al. Improving exploration in evolutionary strategies for deep reinforcement learning via a population of novelty-seeking agents. *Advances in Neural Information Processing Systems (NIPS)* (Curran Associates, Red Hook, 2018).
54. Stanton, C. & Clune, J. Deep curiosity search: Intra-life exploration improves performance on challenging deep reinforcement learning problems. Preprint at https://arxiv.org/abs/1806.00553 (2018).
55. Stanley, K. O. & Miikkulainen, R. A taxonomy for artificial embryogeny. *Artif. Life* **9**, 93–130 (2003).
56. Rawal, A. & Miikkulainen, R. From nodes to networks: evolving recurrent neural networks. Preprint at https://arxiv.org/abs/1803.04439 (2018).
57. Real, E., Aggarwal, A., Huang, Y. & Le, Q. V. Regularized evolution for image classifier architecture search. Preprint at https://arxiv.org/abs/1802.01548 (2018).
58. Dawkins, R. *The Extended Phenotype: The Gene as the Unit of Selection* (Freeman, Oxford, 1982).
59. Gould, S. J. *Full House* (Harvard Univ. Press, Cambridge, 2011).
60. Goldberg, D. E. & Richardson, J. Genetic algorithms with sharing for multimodal function optimization. In *Proc. 2nd International Conference on Genetic Algorithms* 41–49 (L. Erlbaum, Hillsdale, 1987).
61. Mahfoud, S. W. *Niching Methods for Genetic Algorithms*. PhD thesis, Univ. Illinois at Urbana-Champaign (1995).
62. Jong, De, K. A. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, Univ. Michigan (1975).
63. Lehman, J. & Stanley, K. O. Abandoning objectives: evolution through the search for novelty alone. *Evol. Comput.* **19**, 189–223 (2011).
64. Neyshabur, B., Salakhutdinov, R. R. & Srebro, N. Path-SGD: path-normalized optimization in deep neural networks. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)* 2422–2430 (MIT Press, Cambridge, 2015).
65. Radcliffe, N. J. Genetic set recombination and its application to neural network topology optimisation. *Neural Comput. Appl.* **1**, 67–90 (1993).
66. Benson-Amram, S. & Holekamp, K. E. Innovative problem solving by wild spotted hyenas. *Proc. R. Soc. B* **279**, 4087–4095 (2012).
67. Kanter, R. M. *The Change Masters: Binnovation and Entrepreneurship in the American Corporation* (Simon & Schuster, New York, 1984).
68. Mouret, J.-B. & Doncieux, S. Encouraging behavioral diversity in evolutionary robotics: an empirical study. *Evol. Comput.* **20**, 91–133 (2012).
69. Mengistu, H., Lehman, J. & Clune, J. Evolvability search: directly selecting for evolvability in order to study and produce it. In *Proc. Genetic and Evolutionary Computation Conference (GECCO)* 141–148 (ACM, 2016).
70. Gravina, D., Liapis, A. & Yannakakis, G. Surprise search: beyond objectives and novelty. In *Proc. Genetic and Evolutionary Computation Conference (GECCO)* 677–684 (ACM, 2016).

71. Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. A. M. T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evolut. Comput.* **6**, 182–197 (2002).
72. Zitzler, E. & Thiele, L. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Trans. Evolut. Comput.* **3**, 257–271 (1999).
73. Pugh, J. K., Soros, L. B. & Stanley, K. O. Quality diversity: a new frontier for evolutionary computation. *Front. Robot. AI* **3**, 40 (2016).
74. Lehman, J. & Stanley, K. O. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proc. 13th Annual Conference on Genetic and Evolutionary Computation (GECCO)* 211–218 (ACM, 2011).
75. Mouret, J.-B. & Clune, J. Illuminating search spaces by mapping elites. Preprint at https://arxiv.org/abs/1504.04909 (2015).
76. Brant, J. C. & Stanley, K. O. Minimal criterion coevolution: a new approach to open-ended search. In *Proc. Genetic and Evolutionary Computation Conference (GECCO)* 67–74 (ACM, 2017).
77. Hodjat, B., Shahrzad, H. & Miikkulainen, R. Distributed age-layered novelty search. In *Proc. 15th International Conference on the Synthesis and Simulation of Living Systems (Alife XV)* 131–138 (MIT Press, Cambridge, 2016).
78. Huizinga, J., Mouret, J.-B. & Clune, J. Does aligning phenotypic and genotypic modularity improve the evolution of neural networks? In *Proc. Genetic and Evolutionary Computation Conference (GECCO)* 125–132 (ACM, 2016).
79. Meyerson, E. & Miikkulainen, R. Discovering evolutionary stepping stones through behavior domination. In *Proc. Genetic and Evolutionary Computation Conference (GECCO)* 139–146 (ACM, 2017).
80. Nguyen, A., Yosinski, J. & Clune, J. Understanding innovation engines: automated creativity and improved stochastic optimization via deep learning. *Evol. Comput.* **24**, 545–572 (2016).
81. Herculano-Houzel, S. The human brain in numbers: a linearly scaled-up primate brain. *Front. Hum. Neurosci.* **3**, 31 (2009).
82. Venter, J. C. et al. The sequence of the human genome. *Science* **291**, 1304–1351 (2001).
83. Striedter, G. F. *Principles of Brain Evolution* (Sinauer Associates, Sunderland, 2005).
84. Russakovsky, O. et al. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision.* **115**, 211–252 (2015).
85. Turing, A. The chemical basis of morphogenesis. *Phil. Trans. R. Soc. B* **237**, 37–72 (1952).
86. Lindenmayer, A. Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *J. Theor. Biol.* **18**, 280–299 (1968).
87. Bongard, J. C. & Pfeifer, R . Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. In *Proc. Genetic and Evolutionary Computation Conference (GECCO)* 829–836 (Kaufmann, 2001).
88. Hornby, G. S. & Pollack, J. B. Creating high-level components with a generative representation for body-brain evolution. *Artif. Life* **8**, 223–246 (2002).
89. Stanley, K. O. Compositional pattern producing networks: a novel abstraction of development. *Genet. Program. Evol. Mach. Spec. Issue Dev. Syst.* **8**, 131–162 (2007).
90. Meinhardt, H. *Models of Biological Pattern Formation* (Academic, London, 1982).
91. Secretan, J. et al. Picbreeder: a case study in collaborative evolutionary exploration of design space. *Evol. Comput.* **19**, 345–371 (2011).
92. Clune, J. & Lipson, H. Evolving three-dimensional objects with a generative encoding inspired by developmental biology. In *Proc. European Conference on Artificial Life* 144–148 (MIT Press, Cambridge, 2011).
93. Cheney, N, MacCurdy, R, Clune, J. & Lipson, H. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. In *Proc. Genetic and Evolutionary Computation Conference (GECCO)* (ACM, 2013).
94. Nguyen, A., Yosinski, J. & Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2015).
95. Huizinga, J., Stanley, K. O. & Clune, J. The emergence of canalization and evolvability in an open-ended, interactive evolutionary system. *Artif. Life* **24**, 157–181 (2018).
96. Liu, R. et al. An intriguing failing of convolutional neural networks and the coordconv solution. In *Proc. 2018 Conference on Neural Information Processing Systems (NIPS)* (Curran Associates, Red Hook, 2018).
97. Gauci, J. & Stanley, K. O. Autonomous evolution of topographic regularities in artificial neural networks. *Neural Comput.* **22**, 1860–1898 (2010).
98. Stanley, K. O., D'Ambrosio, D. B. & Gauci, J. A hypercube-based indirect encoding for evolving large-scale neural networks. *Artif. Life* **15**, 185–212 (2009).
99. Fernando, C. et al. Convolution by evolution: differentiable pattern producing networks. In *Proc. Genetic and Evolutionary Computation Conference (GECCO)* 109–116 (ACM, 2016).
100. Ha, D., Dai, A. & Le, Q. V. Hypernetworks. In *Proc. 2017 International Conference on Learning Representations* Vol. 2 (OpenReview, 2017).
101. van Steenkiste, S., Koutník, J., Driessens, K. & Schmidhuber, J. A wavelet-based encoding for neuroevolution. In *Proc. Genetic and Evolutionary Computation Conference (GECCO)* 517–524 (ACM, 2016).
102. Koutnik, J., Gomez, F. & Schmidhuber, J. Evolving neural networks in compressed weight space. In *Proc. Genetic and Evolutionary Computation Conference (GECCO)* 619–626 (ACM, 2010).
103. Hausknecht, M., Lehman, J., Miikkulainen, R. & Stone, P. A neuroevolution approach to general atari game playing. *IEEE Trans. Comput. Intell. AI Games* **6**, 355–366 (2014).
104. Turner, A. J. & Miller, J. F. Recurrent cartesian genetic programming of artificial neural networks. *Genet. Program. Evol. Mach.* **18**, 185–212 (2017).
105. Risi, S. & Stanley, K. O. Indirectly encoding neural plasticity as a pattern of local rules. In *Proc 11th International Conference on Simulation of Adaptive Behavior* (Springer, New York, 2010).
106. Risi, S. & Stanley, K. O. An enhanced hypercube-based encoding for evolving the placement, density and connectivity of neurons. *Artif. Life J.* **18**, 331–363 (2012).
107. Schmidhuber, J. *Evolutionary Principles in Self-referential Learning, or on Learning How to Learn: The Meta-meta-...Hook*. PhD thesis, Technische Univ. München (1987).
108. Duan, Y. et al. RL^2: Fast reinforcement learning via slow reinforcement learning. Preprint at https://arxiv.org/abs/1611.02779 (2016).
109. Finn, C, Abbeel, P. & Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. 34th International Conference on Machine Learning* 1126–1135 (PMLR, 2017).
110. Miconi, T. Learning to learn with backpropagation of Hebbian plasticity. Preprint at https://arxiv.org/abs/1609.02228 (2016).
111. Wang, J. X. et al. Learning to reinforcement learn. Preprint at https://arxiv.org/abs/1611.05763 (2016).
112. Floreano, D. & Urzelai, J. Evolutionary robots with on-line self-organization and behavioral fitness. *Neural Netw.* **13**, 431–4434 (2000).
113. Floreano, D. & Mondada, F. Evolution of plastic neurocontrollers for situated agents. *IEEE Trans. Syst. Man. Cybern.* **26**, 396–407 (1996).
114. Hebb, D. O. *The Organization of Behavior: A Neuropsychological Theory* (Wiley, Hoboken, 1949).
115. Soltoggio, A., Bullinaria, A. J., Mattiussi, C., Dürr, P. & Floreano, D. Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios. In *Proc. 11th International Conference on Artificial Life (Alife XI)* (eds Bullock, S. et al.) 569–576 (MIT Press, Cambridge, 2008).
116. Risi, S. & Stanley, K. O. A unified approach to evolving plasticity and neural geometry. In *Proc. International Joint Conference on Neural Networks (IJCNN-2012)* (IEEE, 2012).
117. Tonelli, P. & Mouret, J.-B. On the relationships between generative encodings, regularity, and learning abilities when evolving plastic artificial neural networks. *PLoS One* **8**, e79138 (2013).
118. Barnes, J. M. & Underwood, B. J. 'Fate' of first-list associations in transfer theory. *J. Exp. Psychol.* **58**, 97 (1959).
119. French, R. M. Catastrophic forgetting in connectionist networks. *Trends Cogn. Sci.* **3**, 128–135 (1999).
120. Ellefsen, K. O., Mouret, J.-B., Clune, J. & Bongard, J. C. Neural modularity helps organisms evolve to learn new skills without forgetting old skills. *PLoS Comput. Biol.* **11**, e1004128 (2015).
121. Velez, R. & Clune, J. Diffusion-based neuromodulation can eliminate catastrophic forgetting in simple neural networks. *PLoS One* **12**, e0187736 (2017).
122. Kirkpatrick, J. et al. Overcoming catastrophic forgetting in neural networks. *Proc. Natl Acad. Sci. USA* **114**, 3521–3526 (2017).
123. Zenke, F, Poole, B. & Ganguli, S. Continual learning through synaptic intelligence. In *Proc. 34th International Conference on Machine Learning* 3987–3995 (PMLR, 2017).
124. Miikkulainen, R. et al. Evolving deep neural networks. Preprint at https://arxiv.org/abs/1703.00548 (2017).
125. He, K., Zhang, X., Ren, S. & Sun, J. Identity mappings in deep residual networks. In *European Conference on Computer Vision* 630–645 (Springer, Berlin, 2016).
126. G. Huang,Liu, Z. Van Der Maaten, L. &Weinberger, K. Q. Densely connected convolutional networks. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2017).
127. Szegedy, C., Ioffe, S. & Vanhoucke, V. Inception-v4, Inception-ResNet and the impact of residual connections on learning. In *Proc. 2017 AAAI Conference on Artificial Intelligence* 4278–4284 (AAAI, 2017).
128. Real, E. et al. Large-scale evolution of image classifiers. In *Proc. 34th International Conference on Machine Learning* (eds Precup, D. & Teh, Y. W.) 2902–2911 (PLMR, 2017).

129. Zoph, B., Vasudevan, V., Shlens, J. & Le, Q. V. Learning transferable architectures for scalable image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition* 8697–8710 (IEEE, 2018).
130. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition* 770–778 (IEEE, 2016).
131. Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V. & Le, Q. V. Autoaugment: learning augmentation policies from data. Preprint at https://arxiv.org/abs/1805.09501 (2018).
132. Schmidhuber, J. Deep learning in neural networks: an overview. *Neural Netw.* **61**, 85–117 (2015).
133. Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R. & Schmidhuber, J. LSTM: a search space odyssey. *IEEE Trans. Neural Netw. Learn. Syst.* **28**, 2222–2232 (2017).
134. Melis, G., Dyer, C. & Blunsom, P. On the state of the art of evaluation in neural language models. In *Proc. 2018 International Conference on Learning Representations* (OpenReview, 2018).
135. Zoph, B. & Le, Q. V. Neural architecture search with reinforcement learning. In *Proc. 2017 International Conference on Learning Representations* (OpenReview, 2017).
136. Marcus, M. P., Santorini, B. & Marcinkiewicz, M. A. Building a large annotated corpus of English: the Penn treebank. *Comput. Linguist.* **19**, 313–330 (1993).
137. Caruana, R. Multitask learning. *Mach. Learn.* **28**, 41–75 (1997).
138. Meyerson, E. & Miikkulainen, R. Pseudo-task augmentation: from deep multitask learning to intratask sharing—and back. In *Proc. 35th International Conference on Machine Learning* (PMLR, 2018).
139. Liang, J., Meyerson, E. & Miikkulainen, R. Evolutionary architecture search for deep multitask networks. In *Proc. Genetic and Evolutionary Computation Conference (GECCO)* 466–473 (ACM, 2018).
140. Elsken, T., Metzen, J. H. & Hutter, F. Neural architecture search: a survey. Preprint at https://arxiv.org/abs/1808.05377 (2017).
141. Fernando, C. et al. Pathnet: evolution channels gradient descent in super neural networks. Preprint at https://arxiv.org/abs/1701.08734 (2017).
142. Houthooft, R. et al. Evolved policy gradients. Preprint at https://arxiv.org/abs/1802.04821 (2018).
143. Wang, C., Xu, C., Yao, X. & Tao, D. Evolutionary generative adversarial networks. Preprint at https://arxiv.org/abs/1803.00657 (2018).
144. Jaderberg, M. et al. Population based training of neural networks. Preprint at https://arxiv.org/abs/1711.09846 (2017).
145. Jaderberg, M. et al. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. Preprint at https://arxiv.org/abs/1807.01281 (2018).
146. Eysenbach, B., Gupta, A., Ibarz, J. & Levine, S. Diversity is all you need: learning skills without a reward function. Preprint at https://arxiv.org/abs/1802.06070 (2018).
147. Miconi, T., Clune, J. & Stanley, K. O. Differentiable plasticity: training plastic neural networks with backpropagation. *Proc. International Conference on Machine Learning* 3556–3565 (PMLR, 2018).
148. Mordvintsev, A., Pezzotti, N., Schubert, L. & Olah, C. Differentiable image parameterizations. *Distill* **3**, e12 (2018).
149. Bansal, T., Pachocki, J., Sidor, S., Sutskever, I. & Mordatch, I. Emergent complexity via multi-agent competition. In *Proc. 2018 International Conference on Learning Representations* (OpenReview, 2018).
150. Silver, D. et al. Mastering the game of Go without human knowledge. *Nature* **550**, 354–359 (2017).
151. Paredis, J. Coevolutionary computation. *Artif. Life* **2**, 355–375 (1995).
152. Pollack, J. B., Blair, A. D. & Land, M. Coevolution of a backgammon player. In *Proc. 5th International Workshop on Artificial Life: Synthesis and Simulation of Living Systems (ALIFE-96)* (eds Langton. C. G. & Shimohara, K.) (MIT Press, Cambridge, 1996).
153. Potter, M. A. & De Jong, K. A. Evolving neural networks with collaborative species. In *Proc. 1995 Summer Computer Simulation Conference* 340–345 (Society for Computer Simulation, 1995).
154. Rosin, C. D. & Belew, R. K. Methods for competitive co-evolution: finding opponents worth beating. In *Proc. 1995 International Conference on Genetic Algorithms* 373–381 (Morgan Kaufmann, Burlington, 1995).
155. Cussat-Blanc, S., Harrington, K. & Pollack, J. Gene regulatory network evolution through augmenting topologies. *IEEE Trans. Evolut. Comput.* **19**, 823–837 (2015).
156. Auerbach, J. E. & Bongard, J. C.On the relationship between environmental and morphological complexity in evolved robots. In *Proc. Genetic and Evolutionary Computation Conference (GECCO)* 521–528 (ACM, 2012).
157. Pfeifer, R. & Bongard, J. *How the Body Shapes the Way We Think: A New View of Intelligence* (MIT Press, Cambridge, 2006).
158. Howard, D. et al. Evolving embodied intelligence from materials to machines. *Nat. Mach. Intell.* https://doi.org/10.1038/s42256-018-0009-9 (2019).
159. Stanley, K. O., Lehman, J. & Soros, L. Open-endedness: the last grand challenge you've never heard of. *O'Reilly Online* https://www.oreilly.com/ideas/open-endedness-the-last-grand-challenge-youve-never-heard-of (2017).

Competing interests

The authors declare no competing interests.

Additional information

Reprints and permissions information is available at www.nature.com/reprints.

Correspondence should be addressed to K.O.S. or J.C. or J.L. or R.M.

Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.