

Neuroevolution in Deep Neural Networks: Current Trends and Future Challenges

Edgar Galván and Peter Mooney

Abstract—A variety of methods have been applied to the architectural configuration and learning or training of artificial deep neural networks (DNN). These methods play a crucial role in the success or failure of the DNN for most problems and applications. Evolutionary Algorithms (EAs) are gaining momentum as a computationally feasible method for the automated optimisation and training of DNNs. Neuroevolution is a term which describes these processes of automated configuration and training of DNNs using EAs. While many works exist in the literature, no comprehensive surveys currently exist focusing exclusively on the strengths and limitations of using neuroevolution approaches in DNNs. Prolonged absence of such surveys can lead to a disjointed and fragmented field preventing DNNs researchers potentially adopting neuroevolutionary methods in their own research, resulting in lost opportunities for improving performance and wider application within real-world deep learning problems. This paper presents a comprehensive survey, discussion and evaluation of the state-of-the-art works on using EAs for architectural configuration and training of DNNs. Based on this survey, the paper highlights the most pertinent current issues and challenges in neuroevolution and identifies multiple promising future research directions.

Index Terms—Neuroevolution, Evolutionary Algorithms, Deep Neural Networks, Deep Learning, Machine Learning.

I. INTRODUCTION

DEEP learning algorithms [51], [59], [83], a subset of machine learning algorithms, are inspired by deep hierarchical structures of human perception as well as production systems. These algorithms have achieved extraordinary results in different exciting areas including computer vision [17], speech recognition [52], [102], board games [128] and video games [101], to mention a few examples. The design of deep neural networks (DNNs) architectures (along with the optimisation of their hyperparameters) as well as their training plays a crucial part for their success or failure [93].

Architecture search is an area of growing interest as demonstrated by the large number of scientific works published in recent years. Broadly speaking these works can be classified into one of two categories: evolution-based methods [5], [32], sometimes referred as neuroevolution [38], [149], and reinforcement learning (RL) methods [142]. Other methods falling outside these two categories have also been proposed in the specialised literature including Monte Carlo-based simulations [106], random search [11] and random search with weights prediction [14], hill-climbing [33], grid search [150], Bayesian optimisation [12], [68].

Edgar Galván and Peter Mooney are both with the Naturally Inspired Computation Research Group and with the Department of Computer Science, Maynooth University, Ireland e-mails: edgar.galvan@mu.ie and peter.mooney@mu.ie

RL architecture-search methods started gaining momentum thanks to their impressive results [6], [16], [90], [154], [156], [157], and more recently, EA architecture-search methods started yielding impressive results in the automatic configuration of DNNs architectures [134], [91], [34]. Moreover, it has been reported that neuroevolution requires less computational time compared to RL methods [101], [134], [139].

In their simplest terms a DNN is a feedforward artificial neural network (ANN) with many hidden layers. Each of these layers constitutes a non-linear information processing unit. This simple description encapsulates the incredible capabilities of DNNs. Usually having two or more hidden layers in an ANN qualifies as a DNN. By adding more layers and more units within a layer, a DNN can represent functions of increasing complexity [51].

Evolutionary Algorithms (EAs) [5], [32], also known as Evolutionary Computation systems, are nature-inspired stochastic techniques that mimic basic principles of life. These *automatic* algorithms have been with us for several decades and are highly popular given that they have proven competitive in the face of challenging problems' features such as discontinuities, multiple local optima, non-linear interactions between variables, among other characteristics [31]. They have also proven to yield competitive results in multiple real-world problems against other Artificial Intelligent methods as well as results achieved by human experts [75], [77].

Finding a well-performing architecture is often a very tedious and error-prone process for Deep Learning researchers. Indeed, Lindauer and Hutter [89] remark that there are over 300 works published in the area of Neural Architecture Search [89]. In this work, we focus our attention exclusively in architecture EAs-based search methods in DNNs as well as EAs-based approaches in training DNNs. Particularly, this work considers both landmark EAs, such as Genetic Algorithms [60], Evolution Strategies [13], [118] and Genetic Programming [76]¹ as well as more recent EA variants, such as Differential Evolution [114], NeuroEvolution of Augmenting Topologies [133] and Grammatical Evolution [121]. Furthermore, we consider the main deep learning architectures, as classified by Liu et al. [93], that have been used in neuroevolution, including Autoencoders [24], Convolutional Neural Networks [79], Deep Belief Networks [151], [152] and Restricted Boltzmann Machines [93], [105]. Other deep learning architectures considered in this study include Recurrent Neural Networks [67] and Long Short Term Memory [46].

¹Evolutionary Programming [39] is another landmark EA, but to the best of our knowledge, there are no neuroevolution works using this paradigm.

Previous literature reviews in the area include those conducted by Floreano et al. [38] and Yao [149], carried out more than one and two decades ago, respectively. More recent works include Stanley et al. [132] and Darwish et al. [23]. The former work explains the influence of modern computational power at scale in allowing the grand ambitions of neuroevolution and deep learning from many years ago to be achieved and fulfilled. The latter work delivers a broader and high-level introduction and overview of swarm intelligence and EAs in the optimisation of the hyperparameters and architecture for neural networks in the data analytics domain. In contrast to these works, our paper provides a new contribution to complement these studies by concentrating on the details of configuration and design of neuroevolution approaches in deep learning. We carefully consider how EAs approaches are applied in deep learning and in particular their specific configuration for this purpose.

The goal of our paper is to provide a timely and comprehensive review in neuroevolution in DNNs. The work is aimed at those researchers and practitioners that are either working in the area or are eager to start working in this exciting growing research area. The configuration and design of artificial deep neural networks is error prone, time consuming and difficult. Our paper will highlight the strengths of EAs as a competitive approach to architecture design. We expect this article will attract the attention of researchers in the DL and EA communities to further investigate effective and efficient approaches to addressing new challenges in neuroevolution in DNNs.

Given the specific nature of this paper, an extensive literature review was undertaken. This involved searches in many online databases using a combination of search strategies in order for this review to be methodologically sound. This literature review aims to outline the state-of-the-art and current practice in the domain of neuroevolution by critically evaluating and integrating the findings of all relevant and high-quality individual studies we have found in this area. To establish the extent of existing research and to conduct an exhaustive search representative of all studies that have been conducted on our topic of interest is a major challenge. As the topic of Neuroevolution in Deep Neural Networks straddles several important areas of research in Computer Science: Neural Networks, Machine Learning, and Evolutionary Algorithms there is a need to search widely in order to capture works which appear in one or more of these research areas. We used searches of Google Scholar, IEEE Xplore, ACM Digital Library, ScienceDirect, arXiv, Springer, Citeseer, the archive of Proceedings of Neural Information Processing Systems, and the archive of Proceedings of International Conference on Machine Learning. We strongly believe our paper outlines our estimation of the state-of-the-art in neuroevolution in DNNs at this point in time.

The rest of the paper is organised as follows. Section II provides some background to DL and EAs. Section III discusses how the architectures of DNNs can be evolved efficiently using EA approaches. This moves onto a discussion in Section IV on the training of DNNs with EAs. Section V sets out some of the major challenges and fertile avenues for future work. Finally, the paper closes with some concluding remarks in Section VI.

II. BACKGROUND

A. Deep Neural Networks

The concept of deep learning originated from the study on artificial neural networks (ANNs). An ANN consists of multiple, simple, connected units denominated neurons, each producing a sequence of real-valued activations and, by carefully adjusting the weights, the ANNs can behave as desired. Depending on the problems and the way the neurons are connected, the process of training an ANN may require “long casual chains of computational stages” [124]. Deep learning emerged as a concept from works such as Hinton et al. [59] and has subsequently become a very active research area [93]. A deep learning algorithm is a class of machine learning algorithms using multiple layers to progressively extract higher level features from the raw data input. The term deep then refers specifically to the number of layers through which the raw data is transformed. In deep learning, each subsequent level attempts to learn in order to transform input data into a progressively more abstract and composite representation. Neuroevolution in DNNs has been applied to the development of a wide range of ANNs including, but not limited to, Convolutional Neural networks, Autoencoders, Deep Belief Networks and Recurrent Neural Networks. In the next sections, we summarise these.

1) *Deep Learning Architecture: Convolutional Neural Networks (CNNs)*: CNNs have shown impressive performance in processing data with a grid-like topology. The deep network consists of a set of layers each containing one or more planes. Each unit in a plane receives input from a neighbourhood in the planes of the previous layer. This idea of connecting units to receptive fields dates back to the 1960s with the perceptron and the animal visual cortex organisation discovered by Hubel and Wiesel [63]. The input, such as an image, is convolved with trainable kernels or filters at all offsets to produce feature maps. These filters include a layer of connection weights. Usually, four pixels in a feature map form a group and this is passed through a function, such as sigmoid function or hyperbolic tangent function. These pixels produce additional feature maps in a layer. n planes are normally used in each layer so that n features can be detected. These layers are called convolutional layers. Once a feature is detected, its exact location is less important and convolutional layers are followed by another layer in charge of performing local averaging and sub-sampling operation. Due to the high dimensionality of the inputs, a CNN classifier may cause overfitting. This problem is addressed by using a pooling process, also called sub-sampling or down-sampling, that reduces the overall size of the signal. Normally, the CNN is trained with the usual backpropagation gradient-descent procedure proposed by Lecun et al. [84].

The learning attractive process of a CNN is determined by three key elements: (i) sparse interaction that reduces the computational processing with kernels that are smaller than the inputs, (ii) parameter sharing that refers to learn one set of parameters instead of learning one set at each location, and finally, (iii) equivariance representation that means that whenever the input changes, the output changes in the same manner [51]. CNNs were the first successful *deep learning*

architectures applied to face detection, handwriting recognition, image classification, speech recognition, natural language processing and recommender systems.

The evolution of these CNN architectures has been slow but remarkable. For example, LeNet [84] proposed in the late 1990s and AlexNet [79], proposed a decade later, are very similar with two and five convolutional layers, respectively. Moreover, they also used kernels with large receptive fields in the layer close to the input and smaller filters closer to the output. A major difference is that the latter used rectified linear units as activation function, which became a standard in neuroevolution in designing CNNs. Since 2012, the use of novel and deeper models took off. For example, in 2014, Simonyan and Zisserman [129] won the Imagenet challenge with their proposed 19-layer model known as VGG19. Other networks have been proposed that not only are deeper but use more complex building blocks. For example, in 2015, Szegedy et al. [143] proposed GoogLeNet, also known as Inception, which is a 22-layer network that used inception blocks. In 2015, the ResNet architecture, consisting of the so-called ResNet blocks, proposed by He et al. [58] won the ImageNet challenge. Moreover, multiple CNNs variants have been proposed such as combining convolutions with an autoencoder [65], using RBMs in a CNN [29], to mention a few examples. A description of the variants of this network can be found in [93].

2) *Deep Learning Architecture: Autoencoders (AEs):*

Autoencoders are simple learning circuits which are designed to transform inputs into outputs with the minimum amount of distortion. An autoencoder consists of a combination of an encoder function and a decoder function. The encoder function converts the input data into a different representation and then the decode function converts the new representation back to the original form. AEs attempt to preserve as much information as possible and they provide range-bounded outputs which make them suitable for data pre-processing and iterative architectures such as DNNs [80]. Despite this work appearing in 2020, the authors suggest that there is “still relatively little work exploring the application (of evolutionary approaches to neural architecture search) to autoencoders. In 2012 Baldi [7] argued that while autoencoders “taken center stage in the deep architecture approach” there was still very little theoretical understanding of autoencoders with deep architectures to date. Choosing an appropriate autoencoder architecture in order to process a specific dataset will mean that the autoencoder is capable of learning the optimal representation of the data [18]. Encoding autoencoders within a chromosome representation means that such an approach could be broad enough to consider most autoencoder variations [18]. As an unsupervised feature learning approach, autoencoders attempt to learn a compact representation of the input data whilst retaining the most important information of the representation. This representation is expected to completely reconstruct the original input. This makes initialisation of the autoencoder critical [61]. Whilst autoencoders can induce very helpful and useful representations of the input data they are only capable of handling a single sample and are not capable of modelling the relationship between pairs of samples in the input data.

3) *Deep Learning Architecture: Deep Belief Networks (DBNs):* Deep Belief Networks can be implemented in a number of ways including Restricted Boltzmann Machines (see Section II-A4) and Autoencoders (see Section II-A2). DBNs are well suited to the problem of feature extraction and have drawn “tremendous attention recently” [152]. DBNs, like other traditional classifiers, have a very large number of parameters and require a great deal of training time [19]. When Restricted Boltzmann Machines (RBMs) are stacked together they are considered to be a DBN. The fundamental building blocks of a DBN are RBMs consisting of one visible layer and one hidden layer. When DBNs are applied to classification problems the feature vectors from data samples are used to set the values of the states of the visible variables of the lower layer of the DBN. Then the DBN is trained to generate a probability distribution over all possible labels of the input data. They offer a good solution to learn hierarchical feature representations from data.

4) *Deep Learning Architecture: Other network types:*

In this subsection we introduce some of other popular and well-studied network architectures namely: Recurrent Neural Networks (RNNs), Restricted Boltzmann Machines (RBMs), and Long Short Term Memory (LSTM).

RNNs: In the case of CNNs input is a fixed-length vector and eventually produce a fixed-length vector as output. The number of layers in the CNN determine the amount of computational steps required. Recurrent Neural Networks (RNNs) are more flexible and they allow operation across a sequence of vectors. The connections between the units in the network form a directed cycle and this creates an internal state of the network allowing to exhibit dynamic temporal behaviour. This internal hidden state allows the RNN to store a lot of information about the past efficiently. RNNs are well suited to sequential data prediction and this has seen them being applied to areas such as statistical language modelling and time-series prediction. However, the computational power of RNNs make them very difficult to train. The principal reasons for this difficulty are mainly due to the exploding and the vanishing gradient problems [67]. In theory RNNs can make use of information in arbitrarily long sequences but in reality they are limited to considering look-back at only a few steps.

RBMs: A Restricted Boltzmann Machine (RBM) is a network of symmetrically connected neuron like units which are designed to make stochastic decisions about whether to be on or off. They are an energy-based neural network. In a RBM there are no connections between the hidden units and multiple hidden layers. Learning occurs by considering the hidden activities of a single RBM as the data for training a higher-level RBM [122]. There is no communication or connection between layers and this is where the *restriction* is introduced to a Boltzmann machine. The RBMs are probabilistic models using a layer of hidden binary variables or units to model the distribution of a visible layer of variables. RBMs have been successfully applied to problems involving high dimensional data such as images and text [82]. As outlined by Fischer and Igel [36], RBMs have been the subject of recent research after being proposed as building blocks of multi-layer learning architectures or DBNs. The concept here is that hidden

neurons extract relevant features from the data observations. These features can then serve as input to another RBM. By this so-called stacking of RBMs in this fashion way, a network can learn features from features with the goal of arriving at a high level representation [102].

LSTM: Long-short-term memory (LSTM) networks are a special type of recurrent neural networks capable of learning long-term dependencies. They work incredibly well on a large variety of problems and are currently widely used. LSTMs are specifically designed to avoid the problem of long-term dependencies. The basic unit within the hidden layer of an LSTM network is called a memory block containing one or more memory cells and a pair of adaptive, multiplicative gating units which gate input and output to all cells in the block [46]. In LSTM networks, it was possible to circumvent the problem of the vanishing error gradients in the network training process by method of error back propagation. An LSTM network is usually controlled by recurrent gates called forgetting gates. Errors are propagated back in time through a potentially unlimited number of virtual layers. In this way, learning takes place in LSTM, while preserving the memory of thousands and even millions of time intervals in the past. Network topologies such as LSTM can be developed in accordance with the specifics of the task. Recurrent neural networks (RNNs) with long short-term memory (LSTM) have emerged as an effective and scalable model for several learning problems related to sequential data [53]. Gers and Schmidhuber [47] showed that standard RNNs fail to learn in the presence of time lags exceeding as few as five to ten discrete-time steps between relevant input events and target signals. LSTM are not affected by this problem and are capable of dealing with minimal time lags in excess of 1000 discrete-time steps. In studies such as those by Gers and Schmidhuber [47], LSTM clearly outperforms previous RNNs not only on regular language benchmarks (according to previous research) but also on context-free languages benchmarks.

B. Evolutionary Algorithms

Evolutionary Algorithms (EAs) [5], [32] refer to a set of stochastic optimisation bio-inspired algorithms that use evolutionary principles to build robust adaptive systems. The field has its origins in four landmark evolutionary methods: Genetic Algorithms [60], [49], Evolution Strategies [119], [125], Evolutionary Programming [39] and Genetic Programming [76]. The key element of these algorithms is undoubtedly flexibility allowing the practitioner to use elements from two or more different EAs techniques. This is the reason why the boundaries between these approaches are no longer distinct allowing to have a more holistic EA framework [98] via [37].

EAs work with a population of μ -encoded potential solutions to a particular problem. Each potential solution, commonly known as individual, represents a point in the search space, where the optimum solution lies. The population is evolved by means of genetic operators, over a number of generations, to produce better results to the problem. Each member of the population is evaluated using a fitness function to determine how good or bad the potential solution is in

the problem at hand. The fitness value assigned to each individual in the population probabilistically determines how successful the individual will be at propagating (part of) its code to further generations. Better performing solutions will be assigned higher values (for maximisation problems) or lower values (for minimisation problems).

The evolutionary process is carried out by using genetic operators. Most EAs include operators that select individuals for reproduction, generate new individuals based on the selected individuals and ultimately determine the composition of the individuals in the population at the following generation. Selection, crossover and mutation are key genetic operators used in most EAs paradigms. The selection operator is in charge of choosing one or more individuals from the population based on their fitness values. Multiple selection operators have been proposed. One of the most popular selection operators is tournament selection for its simplicity. The idea is to select the best individual from a pool, normally of size = [2–7], from the population. The stochastic crossover, also known as recombination, operator exchanges material normally from two selected individuals. This operator is in charge of exploiting the search space. The stochastic mutation operator makes random changes to the genes of the individual. This operator is in charge of exploring the search space. The mutation operator is important to guarantee diversity in the population as well as recovering genetic material lost during evolution.

The evolutionary process explained before is repeated until a condition is met. Normally, until a maximum number of generations has been executed. The population in the last generation is the result of exploring and exploiting the search space over a number of generations. It contains the best evolved potential solutions to the problem and may also represent the global optimum solution. Alg. 1 shows the general steps of a EA for a deep CNN network design.

1) Evolutionary Algorithm: Genetic Algorithms (GAs):

This EA was introduced by Holland [60] in the 1970s and highly popularised by Goldberg [49]. This was due to the fact of achieving extraordinary results as well as reaching multiple research communities including machine learning and neural networks. GAs were frequently described as function optimisers, but now the tendency is to consider GAs as search algorithms able to find near-optimal solutions. Multiple forms of GAs have been proposed in the specialised literature. The bitstring fixed-length representation is one of the most predominant encodings used in GAs. Crossover, as the main genetic operator, and mutation as the secondary operator, reproduce offspring over evolutionary search.

2) Evolutionary Algorithm: Genetic Programming (GP):

This EA is a subclass of GAs and was popularised by Koza [76] in the 1990s. GP is a form of automated programming. Individuals are randomly created by using functional and terminal sets, that are required to solve the problem at hand. Even though multiple types of GP have been proposed in the specialised literature, the typical tree-like structure is the predominant form of GP in EAs. Cartesian GP [100] is another form of GP, which has been used in neuroevolution in DNNs [135], [136].

Algorithm 1 A common EA process for network design. Adapted from [148]

- 1: **Input:** the reference dataset D , the number of generations T , the number of individuals in each generation N , the mutation and crossover probabilities P_m and P_c ;
 - 2: **Initialisation:** generating a set of randomised individuals $\{\mathbb{M}_{0,n}\}_{n=1}^N$, and computing their recognition accuracies;
 - 3: **for** $t = 1, 2, \dots, T$ **do**
 - 4: **Selection:** producing a new generation $\{\mathbb{M}'_{t,n}\}_{n=1}^N$ with a Russian roulette process on $\{\mathbb{M}_{t-1,n}\}_{n=1}^N$;
 - 5: **Crossover:** for each pair $(\{\mathbb{M}_{t,2n-1}, \mathbb{M}_{t,2n}\})_{n=1}^{\lfloor N/2 \rfloor}$, performing crossover with probability P_c ;
 - 6: **Mutation:** for each non-crossover individual $\{\mathbb{M}_{t,n}\}_{n=1}^N$, performing mutation with probability P_m ;
 - 7: **Fitness evaluation:** computing the fitness (e.g., recognition accuracy) for each new individual $\{\mathbb{M}_{t,n}\}_{n=1}^N$;
 - 8: **end for**
 - 9: **Output:** a set of individuals in the final generation $\{\mathbb{M}_T, n\}_{n=1}^N$ with their fitness values.
-

3) Evolutionary Algorithm: Evolution Strategies (ES):

These EAs were introduced in the 1960s by Rechenberg [119] and Schwefel [125]. ES are generally applied to real-valued representations of optimisation problems. In ES, mutation is the main operator whereas crossover is the secondary, optional, operator. Historically, there were two basic forms of ES, known as the (μ, λ) -ES and the $(\mu + \lambda)$ -ES. μ refers to the size of the parent population, whereas λ refers to the number of offspring that are produced in the following generation before selection is applied. In the former ES, the offspring replace the parents whereas in the latter form of ES, selection is applied to both offspring and parents to form the population in the following generation. Nowadays, the Covariance Matrix Adaptation-ES, proposed in the 1990s by Hansen [55], [56], [57], is the state of the art ES that adapts the full covariance matrix of a normal search (mutation) distribution.

4) Evolutionary Algorithm: Evolutionary Programming (EP):

These EAs were proposed in the 1960s by Fogel, Owens and Walsh [39] and very little differences are observed between ES and EP. The main difference, however, between these two EAs paradigms is the lack of use of crossover in EP whereas this genetic operator is secondary, and rarely used, in ES. Another difference is that in EP, normally M parents produce M offspring, whereas in ES the number of offspring produced by genetic operators is higher than their parents.

5) Evolutionary Algorithm: Others:

Multiple evolutionary-based algorithms have been proposed in the specialised literature. Relevant to this work are Differential Evolution (DE), Grammatical Evolution (GE) as well as NeuroEvolution of Augmenting Topologies (NEAT).

DE: Differential Evolution (DE) was proposed by Price and Storn [114] in the 1990s. The popularity of this EA is due to the fact that has proven to be highly efficient in continuous search spaces and it is often reported to be more robust as well as achieving a faster convergence speed compared to other optimisation methods [115]. Unlike traditional EAs, the DE-variants perturb the population members with the scaled differences of randomly selected and distinct population members.

GE: Grammatical Evolution (GE) is a grammar-based EA proposed by Ryan et al. [121] in the 1990s. A genotype-phenotype mapping process is used to generate (genetic) programs by using a binary string to select production rules in a Backus-Naur form grammar definition. GE can be seen

as a special form of GP, where one of the main differences is that unlike GP, GE does not perform the evolutionary process on the programs themselves.

NEAT: NeuroEvolution of Augmenting Topologies (NEAT) is a form of EA proposed by Stanley and Miikkulainen [133] in the 2000s. NEAT is a technique for evolving neural networks. Three elements are crucial for NEAT to work: (i) historical marking, that allows solutions to be crossed over, (ii) speciation, that allows for defining niches and (iii) starting from minimal structure, that allows to incrementally find better solutions.

III. EVOLVING DNNs ARCHITECTURES THROUGH EVOLUTIONARY ALGORITHMS

A. The Motivation

In recent years, there has been a surge of interest in methods for neural architecture search. Broadly, they can be categorised in one of two areas: evolutionary algorithms or reinforcement learning. Recently, EAs have stated gaining momentum for designing deep neural networks architectures [34], [38], [74], [91], [99], [116], [117], [133], [148]. The popularity of these algorithms is due to the fact that they are gradient-free, population-based methods that offer a parallelised mechanism to simultaneously explore multiple areas of the search space while at the same time offering a mechanism to escape from local optima. Moreover, the fact that the algorithm is inherently suited to parallelisation means that more potential solutions can be simultaneously computed within acceptable wall-clock time. Steady increase in computing power, including graphics processing units with thousands of cores, will contribute to speed up the computational calculations of population-based EAs.

B. The Critique

Despite the popularity of EAs for designing deep neural network architectures, they have also been criticised in the light of being slow learners as well as being computationally expensive to evaluate [31]. For example, when using a small population-based EA of 20 individuals (potential solutions) and using a training set of 50,000 samples, one generation alone (of hundreds, thousands or millions of generations) will require one million evaluations through the use of a fitness function.

C. Deep Learning Architecture: Convolutional Neural Networks

Agapitos et al. [1] used a tree-based GP system to evolve a hierarchical feature construction as well as a classification system with feedforward processing. Inspired by the work carried out by Bengio et al. [10] where they empirically demonstrated, using DBN, that greedy-unsupervised layer-training strategy helps to optimise deep networks, Agapitos et al. trained a new layer of CNN of image transformation at the time with the goal to learn a stack of gradually better representations. The authors attained good results using the MNIST dataset. Dufourq and Bassett [30] used a GA to evolve CNN architectures. The encoding used by the authors also allowed them to evolve the learning rate denoting the value which is applied during the training optimisation. They used different operations and sizes of filters including one and two-dimension convolution, one and two-dimension max pooling, dropout, among others. They tested their approach in both image classification tasks as well as in sentiment analysis tasks. Chromosomes using two-dimensional convolution were penalised for a text sentiment problem. The authors used a fitness function with two aggregate elements, one denoting the accuracy and the other the complexity of the solution captured by the number of parameters used by a chromosome. The authors reported competitive results compared to state-of-the-art algorithms on the balanced-based and digit-based EMNIST dataset as well as in the Fashion dataset. Desell [28] proposed an algorithm based on NEAT [133] to evolve CNN architectures. Desell carried out some modifications to the NEAT algorithm to evolve CNN architectures through selection, crossover and mutation. Whereas all operators played an important role to produce well-behaved CNNs, it was interesting to see how the use of mutation, involving 7 types of operations, seemed to be crucial to the competitive results reported on the MNIST dataset.

Zoph et al. [157] proposed NASNet search space defined by a predetermined outer structure depicted in Fig. 1 with the ultimate goal of enabling transferability. This structure is composed of convolutional cells, called normal cell (coloured in pink in Fig. 1 (a)) and reduction cell (coloured in grey), repeated many times. The former type of cells returns a feature map of the same dimensions whereas the latter returns a feature map where its height and width is reduced by a factor of two. All cells of the same type are constraint to have the same architecture and it was found beneficial that architectures of normal cells were different to the architectures of reduced cells. The goal of their architecture search process was to discover the architectures of these two types of cells, an example of this is denoted in Fig. 1 (b). In 2019, Real et al. [116] proposed AmoebaNet-A to evolve an image classifier achieving superior accuracy over hand-designed methods for the first time. The authors used a population-based EA with each fixed length member encoding the architecture of CNNs. To do so, they used the NASNet search space [157]. The goal of their EA-based approach was to discover the architectures of the normal cells and the reductions cells as depicted in Fig. 1 (a). Real et al. used a modified version of tournament

selection and two types of mutation as the two genetic operators in charge of driving evolution. Tournament selection (see Section II to read about how this works) was modified so that the newest genotypes were chosen over older genotypes. The mutation operator involved one of two operations taking place once for each individual: the hidden state mutation and the op mutation. To execute any of these types of mutation, first a random cell is chosen, then a pairwise combination is stochastically selected (see Fig. 1 (c)), and finally, one of these two pairs is picked randomly. This hidden state is replaced with another hidden state with the constraint that no loop is formed. The op mutation differs only in modifying the operation used within the selected hidden state. Fig. 1 (d) shows how these two mutation operations work. The authors used the CIFAR-10 dataset to test their proposed AmoebaNet-A and compared it against a reinforcement learning-based method and random search, achieving better accuracy results as well as reducing the computational time required by their algorithm compared to the other two methods. Moreover, the authors used the fittest chromosome found by their algorithm and retrained it using the Imagenet dataset. With this, they also reported encouraging accuracy results compared with other architecture search methods.

In a different constraint setting, Xie et al. [148] proposed Genetic CNN to automatically learn the structures of CNNs with a limited number of layers as well as limited sizes and operations of convolutional filters, to mention a few constraints adopted by them. The authors adopted a GA with binary fixed-length representation to represent parts of evolved network. In their studies, each network is composed by various stages and each of these is composed of nodes that represent convolutional operations. The binary encoding adopted by Xie et al. represents the connection between a number of ordered nodes. This representation allowed the authors to use crossover, along with roulette selection and mutation. They defined a stage as the minimal unit to apply crossover and mutation. This allowed them to maintain the ordered nodes defined in the genotype and produce valid potential solutions only. Even with the restrictions adopted in their work, the authors achieved competitive accuracy results using the CIFAR-10 and MNIST datasets. They also demonstrated how their approach can be generalised by using the learned architecture on the ILSVRC2012 large-scale dataset. This was achieved because their approach was able to produce chain-shaped networks such as AlexNet [79], VGGNet [129], multiple-path networks such as GoogLeNet [143] and highway networks such as Deep ResNet [58], which have been reported to be beneficial when applied to computer vision problems.

Real et al. [117] used an EA to automatically optimise CNN architectures. Individual architectures are encoded as a graph, where the vertices represent rank-3 tensors: two of these represent the spatial coordinates of the image and the third is the number of channels. Activation functions, such as batch-normalisation [64] with rectified linear units (ReLus) or plain linear units, are applied to the vertices. The authors primarily used 11 types of mutations falling into one of three different categories including inserting layers, removing layers as well as using a mutation to modify layers parameters. Although

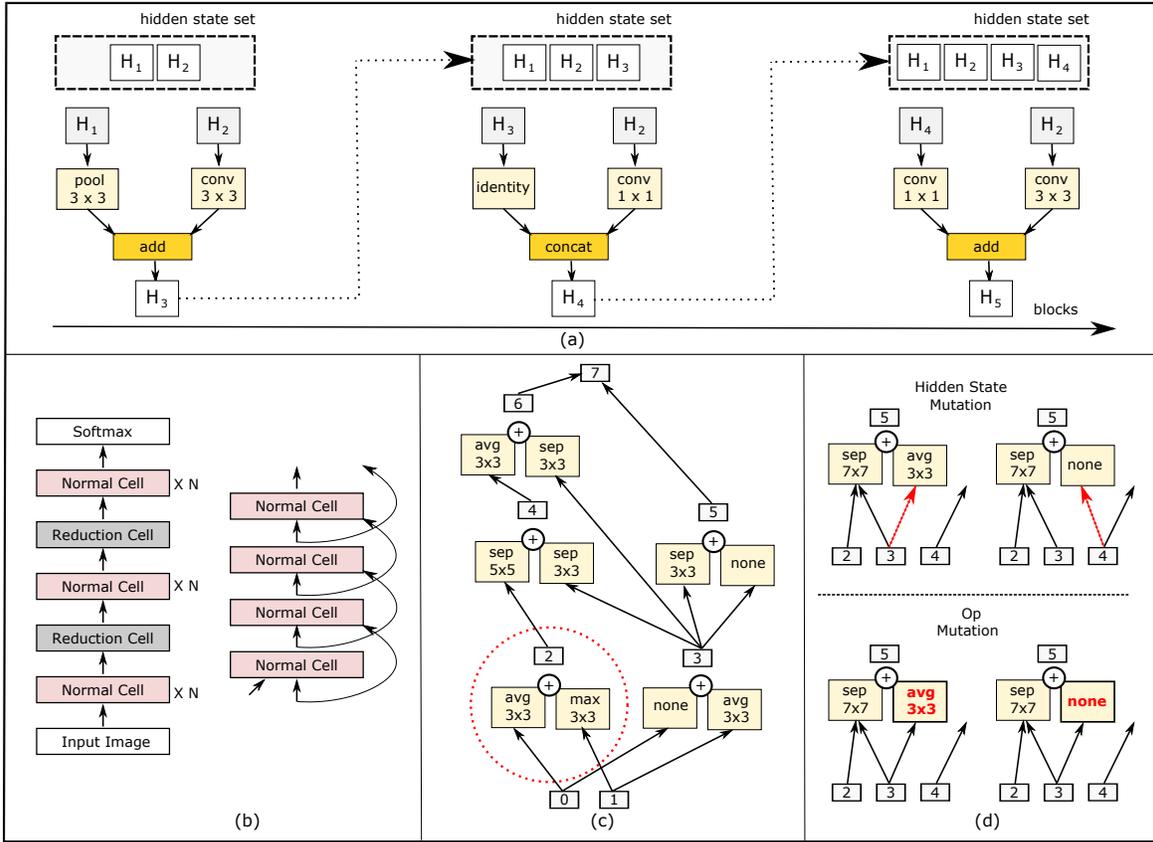


Fig. 1. (a) NASNet Search Space [157]. (b) Scalable architecture for image classification consist of two repeated motifs termed Normal Cell and Reduction Cell. Left: the full outer structure (omitting skip inputs for clarity) and Right: detailed view with the skip inputs. (c) Example of a cell, where the dotted red circle demarcates a pairwise combination. (d) Examples of how the mutations are used. (a) and (b – d) redrawn from Zoph et al. [157] and Real et al. [116], respectively.

they also conducted studies using three forms of crossover, Real et al. indicated that none of these improved the results yielded by mutation operators. The authors also indicated that at the beginning of the search, the EA was susceptible to becoming trapped at local optima. To this end, they applied five mutations per reproduction and decreased this to one at a later stage during evolution. Real et al. used back-propagation to optimise the weights of the CNNs. Because training a large model is incredible slow within their evolutionary setting, the authors partly addressed this by allowing children to inherit their parents' weights when layers had matching shapes. They also used a high-computational setting (250 computers) to carry out their experiments. In their results, the authors reported competitive average accuracy results over five independent runs in the CIFAR-10 and CIFAR-100 datasets compared to state-of-the-art algorithms including ResNet [58] and DenseNet [62].

Suganuma et al. [136] used Cartesian GP [100] (CGP) to automatically design CNN architectures. The genotype encodes information on the type and connections of the nodes. Fig. 2 (a) depicts this idea. These types include ConvBlock, ResBlock, max pooling, average pooling, summation and concatenation. ConvBlock consists of standard convolution processing followed by batch normalisation and ReLU [105] whereas ResBlock is a ConvBlock followed by tensor sum-

mation. The CGP encoding scheme represents the program as a directed acyclic graph in a two-dimensional grid of N_r rows by N_c columns. Fig. 2 (b) provides an example of the phenotype, obtained from Fig. 2 (a), in the case of a grid defined by $N_r = 2$ by $N_c = 3$. The corresponding CNN architecture is depicted in Fig. 2 (c). In their experiments, the authors used the CIFAR-10 dataset as well as a portion of this. As evaluating each of the CGP individuals is expensive, they adopted a simple $(1+\lambda)$ ES (see Section II). The authors' approach achieved competitive results compared with well-known methods including ResNet [58]. It is interesting to see how the authors reported encouraging results using CGP to automatically configure CNN architectures regardless of the sample size used in their work. For example, the CNN architecture produced by CGP for the small dataset scenario is wider compared to the architecture yield by CGP for the standard scenario.

Assunção et al. [3], [4] proposed DENSER, an hybrid mechanism of GAs and (Dynamic Structured) Grammatical Evolution (GE) [121], to evolve DNNs architectures. The outer layer of their proposed approach is in charge of encoding the macro structure of the DNNs evolved by means of GAs. Dynamic Structured GE is in charge of the inner layer that encodes the parameters of the DNNs in a backus-naur form. The authors used the typical genetic operators,

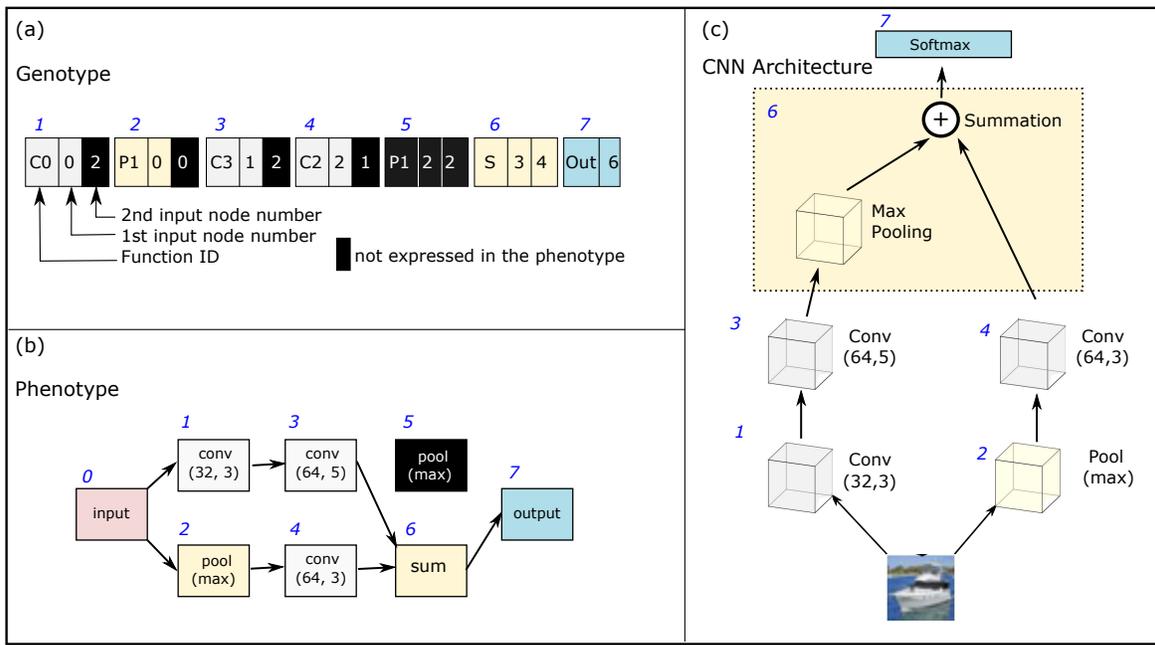


Fig. 2. (a) Genetic representation of a cartesian genetic programming individual encoding a CNN architecture. (b) The phenotypic representation. (c) The CNN architecture defined by (a). Notice that the Gene No. 5, coloured with a black background in the genotype (a) is not expressed in the phenotype. The summation node in (c), highlighted with a light yellow background, performs max pooling to the left-hand side of the input (Node no. 3) to get the same input tensor sizes. Redrawn from Suganuma et al. [136].

including selection, two forms of crossovers (one-point and bit-mask) and three types of mutations (add, replicate and remove unit) in the outer GA-based layer. Furthermore, they used three types of mutations in the inner GE-based layer. To test their approach, the authors used multiple datasets including CIFAR-10, CIFAR-100, MNIST, Fashion-MNIST, SVHN and Rectangles. Similarly to the works conducted by Miikulainen et al. [99] and Suganuma et al. [136], Assunção et al. performed only 10 epochs to train the DNNs due to the high computational cost associated to this. The authors ran 10 runs and reported competitive results compared to state-of-the-art algorithms including using other automatic CNNs designing methods. It is interesting to note that when the authors computed the average fitness and the average number of hidden layers across the entire population, they noticed that these two trends were opposite. That is, as the fitness increases over time, the number of hidden layers decreases over time. This would suggest that these two metrics are in conflict when optimising CNNs architectures.

Sun et al. [141] proposed the use of a population-based GA, of a fixed-length encoding, to evolve, by means of selection, crossover and mutation, unsupervised DNNs for learning meaningful representations for computer vision tasks. Their approach included two main parts (i) finding the optimal architectures in DNNs, the desirable initialisation of weights as well as activation functions, and (ii) fine-tuning all the parameter values in connection weights from the desirable initialisation. The first was primarily achieved by using an encoding, which was inspired by the work conducted by Zhang et al. [153], who captured all of the elements described in (i). As one gene represents on average 1,000 parameters in this encoding, the exploitation achieved by crossover is reduced.

To overcome this problem, Sun et al. used backpropagation in Part (ii). By hand-crafting the various parts of their approach, the authors demonstrated how the local search adopted in Part (i) was necessary in order to achieve promising results.

Recently, Sun et al. [140] proposed a GA, named Evolving Deep CNNs, to automatically discover CNN architectures and corresponding connection weight values. Inspired by the large computational resources reported in the work by Real et al. [117] who used 250 high-end computers, Sun et al. aimed to tackle the use of this resource intensive setting. They proposed a cost-effective method to evaluate the fitness of the individuals in the population allowing them to execute 30 independent runs, as normally adopted by the EA community, in each of the 9 datasets used in their experiments. Moreover, they also used the normal evolutionary operators from EAs, including selection, mutation and crossover. The latter operator was not used in the studies carried out by Real et al. [117]. This was a limitation in Real's work because crossover deals with exploiting the search space. In Sun et al.'s approach each variable-length chromosome encodes the convolutional layer, the pooling layer and the full connection layer. Because hundred of thousands connection weights may exist in one convolutional or full connection layer these should not be directly encoded into the chromosome. Thus, Sun et al. used two statistical measures: the standard deviation and the average value of the connection weights. By doing so, they were able to efficiently evaluate each chromosome in the population. They evaluated each individual by using the classification error as well the number of connection weights. To mitigate the computational time required to evaluate the chromosomes along with the normally CNN deep architectures the authors restricted the training to a small number of epochs (≤ 10).

It is in the last epoch where the fitness is computed for each of the chromosomes in the population. Crossover was applied by using unit alignment, which basically groups parts of the same type, cross them over, and finally, use a restoring phase to put the units back in their corresponding positions in the chromosome. The authors reported highly encouraging results with many cases achieving better results compared to state-of-the-art algorithms in the datasets commonly used by the deep learning community.

van Wyk and Bosman [144] described and evaluated their proposed neural architecture search (NAS) method to automate the process of finding an optimal CNN architecture for the task of arbitrary image restoration. Their work demonstrates the potential feasibility for performing NAS under significant memory and computational time constraints. The ImageNet64x64 dataset was chosen for evaluation. The training set was used for gradient-based optimisation of the NNs performance on unseen data. The authors find that the human-based configured architecture was heavily overparameterised while this was not the case with the evolved NN which performed the tasks with a significantly lower number of total parameters. Sun et al. [139] proposed an encoding strategy built on the state-of-the-art blocks namely ResNet and DenseNet. The authors used a variable length GA that allowed them to automatically evolve CNNs architectures of unrestricted depth. Sun et al. used selection, crossover and mutation to evolve candidate solutions. Given the nature of the variable length encoding used in their approach, the authors employed a repair mechanism that allowed them to produce valid CNNs. The authors used the CIFAR-10 and the CIFAR-100 datasets and compared their results against 9 manually designed methods, 4 semi-automatic methods and 5 automatic methods. Interestingly their results outperformed all hand-crafted methods as well as all semi-automatic methods in terms of the classification error rate.

Evolutionary Multi-objective Optimisation [21], [22], [25], explained in Section V-C, has been hardly used in the automatic configuration of DNNs networks as well as in the optimisation of their hyperparameters. Works on the latter include the recent approach proposed by Kim et al. [71], where the authors used speed and accuracy as two conflicting objectives to be optimised by means of EMO through the use of the Nondominated Sorting Genetic Algorithm II (NSGA-II) [26]. The authors reported interesting results using three classification tasks, including the use of the MNIST, CIFAR-10 and the Drowsy Behaviour Recognition datasets. Inspired by the Kim et al. [71] study, Lu et al. [94] used the same EMO with the same conflicting objectives. It is worth noting that Lu et al. [94] empirically tested multiple computational complexity metrics to measure speed including number of active nodes, number of active connections between nodes and floating-point operations (FLOPs), to mention some. Lu et al. indicated that the latter metric was more accurate and used it as a second conflicting objective for optimisation. Moreover, the authors used an ingenious bitstring encoding in their genetic algorithm which allowed them to use common and robust genetic operators normally adopted in GAs, including homogeneous crossover and bit-flip mutation (at most one change for each mutation operation). The authors

tested their EMO approach with the CIFAR-10 and CIFAR-100 datasets achieving competitive results against state-of-the-art algorithms, including reinforcement learning-based approaches and human expert configurations.

Wang et al. [145] explored the ability of differential evolution to automatically evolve the architecture and hyperparameters of deep CNNs. The method called DECNN uses differential evolution where control of the evolution rate is managed by the a differential value. The DECNN evolves variable-length architectures for CNNs. An IP-based encoding strategy is implemented here to use a single IP address to represent one layer of a DNN. This IP address is then pushed into a sequence of interfaces corresponding to the same order as the layers in DNNs. Six of the MNIST datasets are used for benchmark testing and the DECNN performed very competitively with 12 state-of-the-art competitors over the six benchmarks. Martín et al. [97] proposed EvoDeep which is an EA designed to find the best architecture and optimise the necessary parameters to train a DNN. It uses a Finite-State Machine model in order to determine the possible transitions between different kind of layers, allowing EvoDeep to generate valid sequences of layers, where the output of one layer fits the input requirements of the next layer. It is tested on the benchmark MNIST datasets. The authors report that the high computational resources required to train the DNN means that future work is needed to make the whole process more computationally efficient.

D. Deep Learning Architecture: AutoEncoders

Suganuma et al. [135] used Cartesian Genetic Programming [100], adopting an ES ($1+\lambda$) technique, and using selection and mutation operators only, to optimise DNS architectures for image restoration. To this end, the authors used convolutional autoencoders (CAEs) built upon convolutional layers and skip connections. By optimising the network in a confined search space of symmetric CAEs, the authors achieved competitive results against other methods without the need of using adversarial training and sophisticated loss functions, normally employed for image restoration tasks.

So et al. [130] evolved a transformer network to be used in sequence-to-sequence language tasks. The encoding search space adopted by the authors was inspired by the NASNet search space proposed by Zoph et al. [157], see Fig. 1 (a). This was modified to express characteristics found in state-of-the-art feed-forward seq2seq networks such as the transformer network used in their work. The minimally tuned search space helped them to seed the initial population using a known transformer model. Because computing the fitness of the population, through the use of the negative log perplexity of the validation set, is time consuming, So et al. proposed Progressive Dynamic Hurdles. In essence, the latter allowed promising solutions to be trained using larger training datasets compared to poor potential solutions. The proposed mechanism, using selection and mutation operators only, yielded better results compared to transformer models in the language tasks used in their studies. Hajewski et al. [54] describe an efficient and scalable EA for neural network architecture

search with application to the evolution of deep encoders. Lander and Shang [80] introduce EvoAE an evolutionary algorithm for training autoencoders for DNNs. This proposed methodology is aimed at improving the training time of autoencoders for constructing DNNs. The EvoAE approach searches in the network weight and network structure space of the autoencoders simultaneously allowing for dual optimality searching. Large datasets are decomposed into smaller batches to improve performance. The good performance on large datasets distributed in cloud-based systems could be a major advantage of this approach as a population of high quality autoencoders can be created efficiently.

Luo et al. [95] propose a novel semi-supervised autoencoder called a discriminant autoencoder for application in fault diagnosis. Here, the proposed discriminant autoencoder has a different training process and loss function from traditional autoencoders. In the case of the discriminant autoencoder it is capable of extracting better representations from the raw data provided. A completely different loss function is used and the representations extracted by the discriminant autoencoder can generate bigger differences between the sample classes. The discriminant autoencoder makes full use of labels and feature variables to obtain the optimal representations, based on which the centers of the groups of samples that can be separated as much as possible. Ashfahani et al. [2] propose DEV DAN as a deep evolving denoising autoencoder for application in data stream analytics. DEV DAN demonstrates a proposal of a denoising autoencoder which is a variant of the traditional autoencoder but focused on retrieving the original input information from a noise perturbation. DEV DAN features an open structure where it is capable of initiating its own structure from the beginning without the presence of a pre-configured network structure. DEV DAN can find competitive network architecture compared with state-of-the-art methods on the classification task using ten prominent datasets (including MNIST).

E. Deep Learning Architecture: Deep Belief Networks

In the work by Chen et al. [19] the authors use DBNs to automatically extract features from images. They propose the EFACV (Evolutionary Function Array Classifier Voter) which classifies features from images extracted by a DBN (composed of stacked RBMs). An evolutionary strategy is used to train the EFACV and is mainly used for binary classification problems. For multi-class classification problems it is necessary to have multiple EFACV. The EFACV shows fast computational speed and a reduction in overall training time. Experiments are performed on the MNIST dataset. Liu et al. [92] describe structure learning for DNNs based on multiobjective optimisation. They propose a multiobjective optimisation evolutionary algorithm (MOEA). The DBN and its learning procedure use an RBM to pretrain the DN layer by layer. It is necessary to remove unimportant or unnecessary connections in DNN and move toward discovering optimal DNN connection structure which is as sparse as possible without loss of representation. Experiments based on the MNIST and CIFAR-10 datasets with different training samples indicate that the MOEA approach is effective.

Zhang et al. [151] use DBNs for a prognostic health management system in aircraft and aerospace design. DBNs offer a promising solution as they can learn powerful hierarchical feature representations from the data provided. The authors propose MODBNE (multiobjective deep belief networks ensemble) which is a powerful multiobjective evolutionary algorithm named MOEA based on decomposition. This is integrated into the training of DBNs to evolve multiple DBNs simultaneously with accuracy and diversity as two conflicting objectives in the problem. The DBN is composed of stacked RBMs which are trained in an unsupervised manner. MODBNE is evaluated and compared against a prominent diagnostics benchmarking problem with the NASA turbofan engine degradation problem. In the proposed approach the structural parameters of the DBN are strongly dependent on the complexity of the problem and the number of training samples available. The approach worked outstandingly well in comparison to other existing approaches. GPU-based implementations will be tested in the future for MODBNE to investigate the acceleration of computational processing speed. Zhang et al. [152] consider the problem of cost-sensitive learning methods. The idea of cost-sensitive learning is to assign misclassification costs for each class appropriately. While the authors report that there are very few studies on cost-sensitive DBNs these networks have drawn a lot of attention by researchers recently. Imbalances in the classes in input data is a problem. If there is a disproportionate number of class instances this can affect the quality of the applied learning algorithms. Zhang et al. argue that DBNs are very well placed to handle these type of imbalanced data problems. The ECS-DBN (Evolutionary Cost-Sensitive Deep Belief Network) is proposed to deal with such problems by assigning differential misclassification costs to the data classes. The ECS-DBN is evaluated on 58 popular Knowledge Extraction-based on Evolutionary Learning (KEEL) benchmark datasets.

F. Other networks: LSTM, RRN, RBM

Shinozaki and Watanabe [127] proposes an optimisation strategy for DNN structure and parameters using an EA and a GA. The DNN structure is parameterised by a directed acyclic graph. Experiments are carried out on phoneme recognition and spoken digit detection. All of the experiments were conducted upon a massively parallel computing platform where the experiments were ran using 62 general-purpose computing on graphics processing units (GPGPUs). RBMs are used in the training phase. Ororbia et al. [108] develop an evolutionary algorithm called EXAMM (Evolutionary eXploration of Augmenting Models) which is designed to devolve recurrent neural networks (RNNs) using a selection of memory structures. RNNs are particularly well suited to the task of performing prediction of large-scale real-world time series data. EXAMM was design to select from a large number of memory cell structures and this allowed the evolutionary approach to yield the best performing RNN architecture.

In Peng et al. [110] the authors propose the LSTM (long short-term memory) neural network which is capable of analysing time series over long time spans in order to make

TABLE I

SUMMARY OF EA REPRESENTATIONS, GENETIC OPERATORS, PARAMETERS AND ITS VALUES USED IN NEUROEVOLUTION IN THE DESIGN OF DNNs ARCHITECTURES, ALONG WITH THE DATASETS USED IN VARIOUS STUDIES WITH THEIR CORRESPONDING COMPUTATIONAL EFFORT GIVEN IN GPU DAYS. AUTOMATIC AND SEMI-AUTO(MATIC) REFER TO WORKS WHERE THE ARCHITECTURE HAS BEEN EVOLVED AUTOMATICALLY OR BY USING A SEMI-AUTOMATIC APPROACH, SUCH AS USING A CONSTRAINED SEARCH SPACE, RESPECTIVELY. THE DASH (–) SYMBOL INDICATES THAT THE INFORMATION WAS NOT REPORTED OR IS NOT KNOWN TO US.

Study	EA Method	Representation	Genetic Operators			EAs Parameters' values			Computational Resources	Datasets	GPU days per run	Automatic/Semi-auto	DNNs
			Cross	Mut	Selec	Pop	Gens	Runs					
Agapitos et al. [1]	GP	Variable length	✓	✓	✓	500, 1,000	100	30	–	MNIST	–	Automatic	CNN
Assunção et al. [3], [4]	GAs, GE	Fixed and variable length	✓	✓	✓	100	100	10	Titan X GPUs	CIFAR-10, 3 MNIST variants, Fashion, SVHN, Rectangles, CIFAR-100	–	Automatic	CNN
Charte et al. [18]	GAs, ES, DE	Variable length	✓	✓	✓	150	20	–	1 Nvidia RTX 2080 GPU	CIFAR10, Delicious, Fashion, Glass, Ionosphere, MNIST, Semeion, Sonar, Spect	Limited to 24 hours	Automatic	AE
Chen et al. [19]	EAs	Fixed length	×	✓	✓	$1 + \lambda$, $\lambda=4$	15000	30	–	MNIST	–	Automatic	DBN
Desell [28]	NEAT-based	Variable length	✓	✓	✓	100	–	10	4,500 volunteered PCs	MNIST	–	Semi-auto	CNN
Goncalvez et al. [50]	GAs	Fixed length	×	✓	✓	$1 + \lambda$, $\lambda=4$	gens	runs	No GPUs	4 Binary Class datasets: Cancer, Diabetes, Sonar, Credit	–	Automatic	CNN
Hajewski et al. [54]	EAs	Variable length	✓	✓	✓	$\mu + \lambda = 10$ (λ, μ not specified)	20	20-40	AWS (Nvidia K80 GPU)	STL10	–	Automatic	AE
Kim et al. [71]	EAs	–	–	–	–	50, 40, 60	–	–	60 Tesla M40 GPUs	MNIST, CIFAR-10, Drowsiness Recognition	–	Semi-auto	CNN
Lander et al. [80]	GAs	Variable length	✓	✓	✓	30	50	5	No GPU	MINST	–	Automatic	AE
Liu et al. [92]	EAs	Variable length	✓	✓	✓	100	5000	30	–	MINST, CIFAR-10	–	Automatic	AE,RBM
Lu et al. [94]	GAs	Fixed length	✓	✓	✓	40	30	–	1 NVIDIA 1080Ti	CIFAR-10, CIFAR-100	8 in both	Automatic	CNN
Martín et al. [97] evoDeep	EAs	Fixed Length	✓	✓	✓	$\lambda + \mu = 10$ ($\lambda, \mu = 5$)	20	–	MNIST	Automatic	–	CNN	
Peng et al. [110]	EAs	Fixed length	✓	✓	✓	10	20	–	–	Electricity price	–	Automatic	LSTM
Real et al. [116]	GAs	Variable length	×	✓	✓	100	–	5	450 K40 GPUs	CIFAR-10, ImageNet	3150, 3150	Semi-auto	CNN
Real et al. [117]	GAs	Variable length	×	✓	✓	1000	–	5	250 PCs	CIFAR-10, CIFAR-100	2750, 2750	Automatic	CNN
Shinozaki and Watanabe [127]	GAs, ES	Fixed length	✓	✓	✓	62	30	–	62 GPUs	AURORA2 Spoken Digits Corpus	2.58	Automatic	RBM
So et al. [130]	EAs	Fixed length	×	✓	✓	100	–	–	200 workers with 1 Google TPU V.2 chip	WMT En-De, WMT En-Fr, WMT En-Cs	–	Semi-auto	AE
Suganuma et al. [135]	ES	Fixed length	×	✓	✓	$1+\lambda$, $\lambda = \{1, 2, 4, 8, 16\}$	250	–	4 P100 GPUs	Cars, CelebA, SVNH	12 (inpainting), 16 (denoising)	Automatic	AE
Suganuma et al. [136]	GP, ES	Variable length	×	✓	✓	$1+\lambda$, $\lambda = \{300, 500, 1500\}$	3	–	Multiple PCs, 2 GPUs: GTX 1080, Titan X	CIFAR-10 (2 variants)	27, 27	Automatic	CNN
Sun et al. [140]	GAs	Variable length	✓	✓	✓	100	100	30	1 PC, 2 GTX 1080 GPUs	Fashion, Rectangle (2 variants), Convex Set, MNIST (5 variants)	8 (fashion), 5 (others)	Automatic	CNN
Sun et al. [139]	GAs	Variable length	✓	✓	✓	20	20	5	3 GTX 1080 Ti GPUs	CIFAR-10, CIFAR-100	27, 36	Automatic	CNN
Sun et al. [140]	GAs	Variable length	✓	✓	✓	100	100	30	2 GTX1080 GPUs	Fashion, Rectangle (2 variants), Convex Set, MNIST (5 variants)	8 (Fashion), 5 (others)	Automatic	CNN
Sun et al. [141]	GAs	Fixed length	✓	✓	✓	50	–	30	–	Fashion, Rectangle (2 variants), Convex Set, MNIST (5 variants) CIFAR-10-BW	–	Automatic	CNN
van Wyk and Bosman [144]	EAs	Fixed length	✓	✓	✓	20	20000	1	1 GPU (GTX 1080)	ImageNet64x64	Halted after 2 hrs	Automatic	CNN
Wang et al. [145]	EAs	Variable length	✓	✓	✓	30	20	30	–	MNIST (5 variants) and Convex Set	–	Automatic	CNN
Xie et al. [148]	GAs	Fixed length	✓	✓	✓	20	50	–	10 GPUs (type not specified)	CIFAR-10, MNIST, ILSVRC2012, SHVN	17 (CIFAR-10), 2 (MNIST), 20 (ILSVRC2012), – (SHNV)	Semi-auto	CNN
Zhang et al. [151]	EAs	Variable length	×	✓	✓	20	500	10	No GPUs	NASA C-MAPSS (Aircraft Engine Simulator Datasets)	–	Automatic	DBN
Zhang et al. [152]	EAs	Fixed Length	✓	✓	✓	–	30	10	1 NVIDIA GTX 980 GPU	58 Knowledge Extraction based on Evolutionary Learning (KEEL) datasets	–	Automatic	DBN

predictions and effectively tackle the vanishing gradient problem. Their study uses differential evolution (DE) to identify the hyperparameters of the LSTM. DE approaches have been shown to outperform other approaches such as particle swarm optimisation and GAs. The authors claim that this is the first time that DE has been used to choose hyperparameters for LSTM for forecasting applications. As forecasting involves complex continuous nonlinear functions, the DE approach is well suited to these types of problems. Goncalvez et al. [50] introduce a neuroevolution algorithm called the Semantic Learning Machine (SLM) which has been shown to outperform

other similar methods in a wide range of supervised learning problems. SLM is described as a geometric semantic hill climber approach for NNs following a $1 + \lambda$ strategy. In the search for the best NN architecture configuration this allows the SLM to concentrate on the current best NN without drawing any penalties for this. The crucial aspect of the SLM approach is the geometric semantic mutation which takes a parent NN and generates a child NN.

G. Final Comments

The use of evolution-based method in designing deep neural networks is already a reality as discussed in this section. Different EAs methods with different representations have been used, ranging from landmark evolutionary methods including Genetic Algorithms, Genetic Programming and Evolution Strategies up to using hybrids combining, for example, the use of Genetic Algorithms and Grammatical Evolution. In a short period of time, we have observed both ingenious representations and interesting approaches achieving extraordinary results against human-based design networks [116] as well as state-of-the-art approaches, in some case employing hundred of computers [117] to using just a few GPUs [140]. We have also learnt that most of the neuroevolution studies has focused their attention in designing deep CNNs. Other networks have also been considered including AE, RBM, RNN, LSTM and DBM, although there are just a few neuroevolution works considering the use of these types of networks.

Table I contains extracted information from almost 30 selected papers in neuroevolution. We selected these papers in our own ad-hoc way in order to find a selection of papers which succinctly demonstrated the use of neuroevolution in deep neural networks. The table is order in alphabetically order of the lead-author surname and summarises: the EA representation used, the representation of individuals, which genetic operators are used, and the EA parameters. The table also outlines the computational resources used in the corresponding study by attempting to outline the number of GPUs used. A calculation of the GPU days per run is approximated in the same way as Sun et al. [139]. The table states which benchmark datasets are used in the experimental analysis. Finally, the table indicates if the neural network architecture has been evolved automatically or by using a semi-automated approach whilst also indicating the target DNN architecture. Every selected paper does not report the same information which some papers omitting details about the computation resources used and others omitting information about the number of runs performed. One of the very interesting outputs from this table is that there are numerous differences between the approaches used by all of the papers listed in the table. We see crossover being omitted from several studies mostly due to encoding adopted by various researchers. Population size and selection strategies for the EAs change between studies. While MNIST and CIFAR are clearly the most popular benchmark datasets we can see many examples of studies using benchmark datasets from specific application domains.

IV. TRAINING DEEP NEURAL NETWORKS THROUGH EVOLUTIONARY ALGORITHMS

A. Motivation

Backpropagation has been one of the most successful and dominant methods used in the training of ANNs over the past number of decades [120]. This simple, effective and elegant method applies Stochastic Gradient Descent (SGD) to the weights of the ANN where the goal is to keep the overall error as low as possible. However, as remarked by Morse and Stanley [104] the widely held belief, up to around 2006,

was that backpropagation would suffer loss of its gradient within DNNs. This turned out to be a false assumption and it has subsequently been proved that backpropagation and SGD are effective at optimising DNNs even when there are millions of connections. Both backpropagation and SGD benefit from the availability of sufficient training data and the availability of computational power. In a problem space with so many dimensions the success of using SGD in DNNs is still surprising. Practically speaking, SGD should be highly susceptible to local optima [104]. EAs perform very well in the presence of saddle points as was discussed in Section II.

B. The critique

As there are no guarantees of convergence the solutions computed using EAs are usually classified as near optimal. Population-based EAs are in effect an approximation of the gradient as this is estimated from the individuals in a population and their corresponding objectives. On the other hand, SGD computes the exact gradient. As a result some researchers may consider EAs unsuitable for DL tasks for this reason. However, it has been demonstrated that the exact approximation obtained by SGD is not absolutely critical in the overall success of DNNs using this approach. Lillicrap et al. [88] demonstrated that breaking the precision of the gradient calculation has no negative or detrimental effect on learning. Indeed, Morse and Stanley [104] speculated that the reason for the lack of research focus on using evolutionary computation in DNNs was not entirely related to concerns around the gradient. It more than likely resulted from the belief that new approaches to DNN could emerge from outside of SGD.

C. Deep Learning Architecture: Convolutional Neural Networks

Such et al. [134] proposed a gradient-free method to evolve the weights of convolutional DNNs by using a simple GA, with a population of chromosomes of fixed length. The proposed mechanism successfully evolved networks with over four million free parameters. Some key elements in the study conducted by Such et al. to successfully evolve these large neural networks include (i) the use of the selection and mutation genetic operators only (excluding the use of the crossover operator), (ii) the use of a novel method to store large parameter vectors compactly by representing each of these as an initialisation seed plus the list of the random seeds that produces the series of mutations that produced each parameter vector, (iii) the use of a state-of-the-art computational setting, including one modern computer with 4 GPUs and 48 CPU cores as well as 720 CPU cores across dozens of computers. Instead of using a reward-based optimisation techniques by means of a fitness function, Such et al. used novelty search [86] that rewards new behaviours of individuals. The authors used reinforcement learning benchmark problems including atari 2600 [9], [101], hard maze [87] and humanoid locomotion [15]. They demonstrated how their proposed approach is competitive with state-of-the-art algorithms in these

problems including DQN [101], policy-gradient methods [126] and ES [123].

Pawelczyk et al. [109] focused their attention in encoding CNNs with random weights using a GA, with the main goal to let the EA to learn the number of gradient learning iterations necessary to achieve a high accuracy error using the MNIST dataset. It was interesting to observe that their EA-based approach reported the best results with around 450 gradient learnt iterations compared to 400 constant iterations which yielded the best overall results.

D. Deep Learning Architecture: Autoencoders

David and Greental [24] used a GA of fixed length to evolve the weight values of an autoencoder DNN. Each chromosome was evaluated by using the root mean squared error for the training samples. In their experiments, the authors used only 10 individuals with a 50% elitism-policy. The weights of these individuals were updated using backpropagation and the other half of the population were randomly generated in each generation. They tested their approach with the well-known CIFAR-10 dataset. They compared their approach *vs.* the traditional autoencoder using SVM, reporting a better classification error when using their proposed GA-assisted method for the autocoder DNN (1.44% *vs.* 1.85%). In their studies, the authors indicated that the reason why their method produced better results was because gradient descent methods such as backpropagation are highly susceptible to being trapped at local optima and their GA method helped to prevent this.

Fernando et al. [35] introduced a differentiable version of the Compositional Pattern Producing Network (CPPN) called the Differentiable Pattern Producing Network (DPPN). The DPPN approach attempts to combine the advantages and results of gradient-based learning in NN with the optimisation capabilities of evolutionary approaches. The DPPN has demonstrated superior results for the benchmark dataset MNIST. A generic evolutionary algorithm is used in the optimisation algorithm of DPPN. The results indicate that the DPPNs and their associated learning algorithms have the ability to dramatically reduce the number of parameters of larger neural networks. The authors argue that this integration of evolutionary and gradient-based learning allows the optimisation to avoid becoming stuck in local optima points or saddle points.

E. Other Relevant Works

Morse and Stanley [104] proposed an approach called limited evaluation evolutionary algorithm (LEEA), that effectively used a population-based GA of fixed length representation to evolve, by means of crossover and mutation, 1000 weights of a fixed-architecture network. The authors took inspiration from SGD that can compute an error gradient from a single (or small batch of) instance of the training set. Thus, instead of computing the fitness of each individual in the population using the whole training set, the fitness is computed using a small fraction. This results in an EA that computationally similar to SGD. However, using such approach is also one of the weakness in LEAA because it does not generalise to whole training sample. To mitigate

this, the authors proposed two approaches: (i) the use of a small batch of instances and (ii) the use of a fitness function that consider both the performance on the current mini-batch and the performance of individuals' ancestors against their mini-batches. To test their idea, the authors used a function approximation task, a time series prediction task and a house price prediction task and compared the results yield by their approach against SGD and RMSProp. They showed how their LEEA approach was competitive against the other approaches. Even when the authors do not use DNNs, but a small artificial NN, it is interesting to note how this can be used in a DNN setting.

Khadka and Tumer [70] remark that Deep Reinforcement Learning methods are "notoriously sensitive to the choice of their hyperparameters and often have brittle convergence properties". These methods are also challenged by long time horizons where there are sparse rewards. EAs can respond very positively to these challenges where the use of fitness metrics allows EAs to tolerate the sparse reward distribution and endure long time horizons. However, EAs can struggle to perform well when optimisation of a large number of parameters is required. The authors introduce their Evolutionary Reinforcement Learning (ERL) algorithm. The EA is used to evolve diverse experiences to train an RL agent. These agents are then subjected to mutation and crossover operators to create the next generation of agents. From the results outlined in the paper this ERL can be described as a "population-driven guide" that guides or biases exploration towards states with higher and better long-term returns, promoting diversity of explored policies, and introduces redundancies for stability.

Recurrent Neural Networks (RNNs) (see Section II-A4) incorporate memory into an NN by storing information from the past within the hidden states network. In [69], Kahdka et al. introduce a new memory-augmented network architecture called the Modular Memory Unit (MMU). This MMU disconnects the memory and central computation operations without requiring costly memory management strategies. Neuroevolutionary methods are used to train the MMU architecture. The performance of the MMU approach with both gradient descent and neuroevolution are examined in the paper. The authors find that neuroevolution is more repeatable and generalizable across tasks. The MMU NN is designed to be highly configurable and this characteristic is exploited by the neuroevolutionary algorithm to evolve the network. Population size is set to 100 with a fraction of elites set at 0.1. In the fully differentiable version of the MMU gradient descent performs better for Sequence Recall tasks than neuroevolution. However, neuroevolution performs significantly better than gradient descent in Sequence Classification tasks.

F. Final Comments

In the early years of neuroevolution, it was thought that evolution-based methods might exceed the capabilities of backpropagation [149]. As ANNs, in general, and as DNNs, in particular, increasingly adopted the use of stochastic gradient descent and backpropagation, the idea of using EAs for training DNNs instead has been almost abandoned by the

TABLE II

SUMMARY ON EA REPRESENTATIONS, GENETIC OPERATORS, PARAMETERS AND ITS VALUES USED IN NEUROEVOLUTION IN THE TRAINING OF DNNs, ALONG WITH THE DATASETS USED IN VARIOUS STUDIES WITH THEIR CORRESPONDING COMPUTATIONAL EFFORT GIVEN IN GPU DAYS. THE DASH (–) SYMBOL INDICATES THAT THE INFORMATION WAS NOT REPORTED OR IS NOT KNOWN TO US.

Study	EA Method	Representation	Genetic Operators			EAs Parameters' values			Computational Resources	Datasets	GPU days per run	DNN
			Cross	Mut	Selec	Pop	Gens	Runs				
David and Greental [24]	GAs	Fixed length	✓	✓	✓	10	–	–	–	MNIST	–	AE
Dufourq and Bassett [30]	GAs	Variable length	×	✓	✓	100	10	5	1 GTX1070 GPU	CIFAR-10, MNIST, EMNIST (Balanced & Digits), Fashion, IMDB, Electronics	–	CNN
Fernando et al. [35]	GAs	–	✓	✓	✓	50	–	–	–	MNIST, Omniglot	–	AE
Khadka and Tumer [70]	EAs	Variable length	✓	✓	✓	10	∞	5	–	6 Mujoco (continuous control) datasets	–	Read text
Khadka et al. [69]	EAs	Fixed length	×	✓	✓	100	1000 10000 15000	–	GPU used but not specified	Sequence Recall, Sequence Classification	–	Read text
Morse and Stanley [104]	GAs	Fixed length	✓	✓	✓	1,000	–	10	–	Function Approximation, Time Series, California Housing	–	Read text
Pawelczyk et al. [109]	GAs	Fixed length	✓	✓	✓	10	–	–	1 GPU (Intel Core i7 7800X, 64GB RAM)	MNIST	–	CNN
Such et al. [134]	GAs	Fixed length	×	✓	✓	1,000 (A), 12,500 (H), 20,000 (L)	–	5 (A), 10 (L)	1 PC (4 GPUs, 48 CPUs) and 720 CPUs across dozens of PCs	Atari 2600, Image Hard maze, Humanoid locomotion	0.6 (Atari, 1 PC), 0.16 (Atari, dozens of PCs)	CNN

DNN research community. EAs are a “genuinely different paradigm for specifying a search problem” [104] and provide exciting opportunities for learning in DNNs. When comparing neuroevolutionary approaches to other approaches such as gradient descent, authors such as Khadka et al. [69] urge caution. A generation in neuroevolution is not readily comparable to a gradient descent epoch.

Despite the fact that it has been argued that EAs can compete with gradient-based search in small problems as well as using neural networks with a non-differentiable activation function [96], the encouraging results achieved in the 1990s [48], [103], [113] have inspired recently some researchers to carry out research in training DNNs including the works conducted by David and Greental [24] and Fernando et al. [35] both works using deep AE as well as the works carried out by Pawelczyk et al. [109] and Such et al. [134], both studies using deep CNNs.

Table II is structured in a similar way to Table I. As with Table I, we selected these papers in our own ad-hoc way in order to find a selection of papers which succinctly demonstrated the use of EAs in the training of DNNs. As before we see mutation and selection used by all of the selected works with crossover omitted in certain situations. We see greater diversity in the types of benchmark datasets used with a greater focus on domain-specific datasets and problems.

V. FUTURE WORK ON NEUROEVOLUTION IN DEEP NEURAL NETWORKS

A. Surrogate-assisted EAs in DNNs

EAs have successfully been used in automatically designing artificial DNNs, as described throughout the paper, and multiple state-of-the-art algorithms have been proposed in recent years including genetic CNN [148], large-scale evolution [117], evolving deep CNN [140], hierarchical evolution [91], to mention but a few successful examples. Despite their success in automatically configuring DNNs architectures, a common limitation in all these methods is the training time needed, ranging from days to weeks in order to achieve

competitive results. Surrogate-assisted, or meta-model based, evolutionary computation uses efficient models, also known as meta-models or surrogates, for estimating the fitness values in evolutionary algorithms [66]. Hence, a well-posed surrogate-assisted EC considerably speeds up the evolutionary search by reducing the number of fitness evaluations while at the same time correctly estimating the fitness values of some potential solutions.

The adoption of this surrogate-assisted EA is limited in the research discussed in this paper and is dealt with in a few limited exceptions. For example, in a recent work, Sun et al. [137] demonstrated how meta-models, using ensemble members, can be successfully used to correctly estimate the accuracy of CNNs. They were able to considerably reduce the training time, e.g., from 33 GPU days to 10 GPU days, while still achieving competitive accuracy results compared to state-of-the-art algorithms. A limitation in Sun et al.’s approach is the unknown number of training runs that is necessary to achieve a good prediction performance.

B. Mutations and the neutral theory

We have seen that numerous studies have used selection and mutation only to drive evolution in automatically finding a suitable DNN architecture (Section III) or to train a DNN (Section IV). Tables I and II present a summary of the genetic operators used by various researchers. Interestingly, a good number of researchers have reported highly encouraging results when using these two genetic operators, including the works conducted by Real et al. [116], [117] using GAs and hundreds of GPUs as well as the work carried out by Suganuma et al. [136] employing Cartesian Genetic Programming and a using a few GPUs.

Kimura’s neutral theory of molecular evolution [72], [73] states that the majority of evolutionary changes at molecular level are the result of random fixation of *selectively neutral mutations*. A mutation from one gene to another is neutral if it does not affect the phenotype. Thus, most mutations that take place in natural evolution are neither advantageous

nor disadvantageous for the survival of individuals. It is then reasonable to extrapolate that, if this is how evolution has managed to produce the amazing complexity and adaptations seen in nature, then neutrality should aid also EAs. However, whether neutrality helps or hinders the search in EAs is ill-posed and cannot be answered in general: one can only answer this question within the context of a specific class of problems, (neutral) representation and set of operators [40], [41], [42], [43], [44], [45], [111], [112]. We are not aware of any works in neuroevolution in DNN on neutrality, but there are some interesting encodings adopted by researchers including Suganuma's work [136] (see Fig. 2) that allow the measurement of the level of neutrality present in evolutionary search and indicate whether its presence is beneficial or not in certain problems and DNNs. If neutrality is beneficial, taking into consideration specific class of problems, representations and genetic operators, this can also have an immediate positive impact in the training time needed because the evaluation of potential EA candidate solutions will not be necessary.

C. Multi-objective Optimisation

The vast majority of works reviewed in this paper have focused their attention in the direct or indirect optimisation of one objective only. For example, when training a CNN in a computer vision supervised classification task, the classification error is normally adopted as a metric of performance for this type of network. Perhaps, one of the reasons why taking into account one objective has been the norm in the specialised literature is because the optimisation of one objective has been enough to yield extraordinary results (for example in the application domain of route optimisation [8]). Another potential reason could be due to the fact that two or more objectives can be conflicting with each other making the (optimisation) task very difficult to accomplished [21], [22], [25].

Multi-objective optimisation (MO) is concerned with the simultaneous optimisation of more than one objective function. When such functions are in conflict, a set of trade-off solutions among the objectives is sought as no single global optimum exists. The optimal trade-offs are those solutions for which no objective can be further improved without degrading one of the others. This idea is captured in the Pareto dominance relation: a solution x in the search space is said to *Pareto-dominate* another solution y if x is at least as good as y on all objectives and strictly better on at least one objective. This is an important aspect in EMO (Evolutionary MO) [21], [22], [25] because it allows solutions to be ranked according to their performance on all objectives with respect to all solutions in the population. EMO is one of the most active research areas in EAs. Yet it is surprising to see that EMO approaches have been scarcely used for the automatic configuration of artificial DNNs architectures or learning in DNNs. Often, the configuration of these artificial DNNs require simultaneously satisfying multiple objectives such as reducing the computational calculation of these on the training dataset while attaining high accuracy. EMO offers an elegant and efficient framework to handle these conflicting objectives. We are aware of only a few works in the area e.g., [71], [92], [94], [151], as summarised in Section III.

D. Fitness Landscape Analysis of DNNs and Well-posed Genetic Operators

As we have seen throughout the paper, all of the works in neuroevolution in DNNs have used core genetic operators including selection and mutation. Crossover has also been used in most of these works. The use of these operators are summarised in Tables I and II. The use of crossover, sometimes referred as recombination, can sometimes be difficult to adopt depending on the encoding used and some variants have been proposed such as in the study carried out in [140]. Other studies have adopted standard crossover operators such as those discussed in [71]. There are, however, no works carried out in the area of neuroevolution in DNNs that have focused their attention in explaining why the adoption of a particular genetic operator is well-suited for that particular problem.

The notion of fitness landscape [146] has been with us for several decades. It is a non-mathematical aid that has proven to be very powerful in understanding evolutionary search. Viewing the search space, defined by the set of all potential solutions, as a landscape, a heuristic algorithm such as an EA, can be thought of as navigating through it to find the best solution (essentially the highest peak in the landscape). The height of a point in this search space, represents in an abstract way, the fitness of the solution associated with that point. The landscape is therefore a knowledge interface between the problem and the heuristic-based EA. This can help researchers and practitioners to define well-behaved genetic operators, including mutation and crossover, over the connectivity structure of the landscape.

E. Standardised Scientific Neuroevolution Studies in DNNs

As described in Section II, multiple DNNs architectures have been proposed in the specialised literature including CNNs, DBNs, RBMs and AEs. Each of these DNNs considers multiple elements such as the activation function, type of learning, to mention a few examples. As indicated previously, EAs are incredible flexible allowing researchers to use elements from two or more different EAs methods. Moreover, multiple variants from each of these elements exists such as having multiple options from where to chose to exploit and explore the search space. Many of the research works reviewed in this paper have compared their results with those yield by neuroevolution-based state-of-the-art algorithms. However, it is unclear why some techniques are better than others. Is it because of the type of operators used? Is it because of the representation adopted in these studies or is it because of the type of learning employed during training? Due to the lack of standardised studies in neuroevolution on DNNs, it is difficult to draw final conclusions that help us to identify what elements are promising in DNNs.

F. Diversifying the use of benchmark problems and DNNs

There is little argument that the availability of new and large datasets combined with ever increasingly powerful computational resources have allowed DNNs to tackle and solve hard problems in domains such as image classification, speech

TABLE III
COMMON DATASETS USED IN NEUROEVOLUTION IN DEEP NEURAL NETWORKS.

Data set	Number of examples		Input Size	RGB, B&W, Grayscale	No. of classes
	Training	Testing			
MNIST [85]	60,000	10,000	28×28	Grayscale	10
MNIST variants [81]	12,000	50,000	28×28	Grayscale	10
CIFAR-10 [78]	50,000	10,000	32×32	RGB	10
CIFAR-100 [78]	50,000	10,000	32×32	RGB	100
Fashion [147]	60,000	10,000	28×28	Grayscale	10
SVHN [107]	73,257	26,032	32×32	RGB	10
Rectangle [81]	1,000	50,000	28 × 28	B&W	2
Rectangle images [81]	10,000	50,000	28 × 28	Grayscale	2
Convex set [81]	6,000	50,000	28 × 28	B&W	2
ILSVRC2012 [27]	1.3M	150,000	224 × 224	RGB	1,000
GERMAN Traffic Sign Recognition [131]	50,000	12,500	32×32	Grayscale	43
CelebFaces [138]	–	–	39 × 31	RGB	2

(No. of images of CelebFaces: 87,628)
Number of examples for the validation set is omitted from this table given that it is well-known in the ML community that this is randomly split from the training data with the proportion of $\frac{1}{5}$.

processing and many others. Image classification is certainly considered as the primary benchmark against which to evaluate DNNs [155]. These benchmark datasets (many of which are outlined in Table III) are used as a means of comparing the computational results of experimental setups created by different research groups. We believe that the success of DNNs coupled with the need to tackle complex problems in other domains sees a growing need for DNNs to expand to other domains. In order to assess how successful DNNs are in other domains and with other practical problems robust and comprehensive benchmark datasets will be required. Indeed we believe that without such benchmarks it may be difficult to make convincing arguments for the success and suitability of DNNs for problems in other domains beyond image classification, machine translation, and problems involving object recognition.

It is critical that benchmark datasets are available freely and as open-data. Stallkamp et al. [131] argue that in a niche area such as traffic sign recognition it can be difficult to compare published work because studies are based on different data or consider classification in different ways. The use of proprietary data in some cases, which is not publicly available, makes comparison of results difficult. Authors such as Zhang et al. [151] access data from a prognostic benchmarking problem related to NASA and Aero-Propulsion systems. Specific problem domains outside of those of vision, speech recognition and language also have benchmark datasets available but may be less well-known. Zhang et al. [152] use datasets from KEEL (Knowledge Extraction based on Evolutionary Learning) but also use a real-world dataset from a manufacturing drilling machine in order to obtain a practical evaluation. Chen and Li [20] comment that as we see data getting bigger (so called Big Data) deep learning will continue to play an ever increasingly important role in providing big data predictive analytics solutions, particularly with the availability of increased processing power and the advances in graphics processors. However, while the potential of Big Data is without doubt, new ways of thinking and novel algorithmic approaches will be required to deal with the technical challenges. Algorithms that can learn from massive amounts of data are needed [20] and this may make it difficult to define benchmark datasets within the Big Data domain.

VI. CONCLUSIONS

This paper has provided a comprehensive survey of neuroevolution approaches in Deep Neural Networks (DNNs) and has discussed the most important aspects of application of Evolutionary Algorithms (EAs) in deep learning. The target audience of this paper is a broad spectrum of researchers and practitioners from both the Evolutionary Computation and Deep Learning (DL) communities. The paper highlights where EAs are being used in DL and how DL is benefiting from this. Readers with a background in EAs will find this survey very useful in determining the state-of-the-art in neural architecture search methods in general. Additionally, readers from the DL community will be encouraged to consider the application of EAs approaches in their DNN work. Configuration of DNNs is not a trivial problem. Poorly or incorrectly configured networks can lead to the failure or under-utilisation of DNNs for many problems and applications. Finding well-performing architectures is often a very tedious and error-prone process. EAs have been shown to be a competitive and successful means of automatically creating and configuring such networks. Consequently, neuroevolution has great potential to provide a strong and robust toolkit for the DL community in future. The article has also outlined and discussed important issues and challenges in this area.

REFERENCES

- [1] A. Agapitos, M. O'Neill, M. Nicolau, D. Fagan, A. Kattan, A. Brabazon, and K. Curran. Deep evolution of image representations for handwritten digit recognition. In *IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015*, pages 2452–2459. IEEE, 2015.
- [2] A. Ashfahani, M. Pratama, E. Lughofer, and Y.-S. Ong. Devdan: Deep evolving denoising autoencoder. *Neurocomputing*, 390:297 – 314, 2020.
- [3] F. Assunção, N. Lourenço, P. Machado, and B. Ribeiro. Evolving the topology of large scale deep neural networks. In M. Castelli, L. Sekanina, M. Zhang, S. Cagnoni, and P. García-Sánchez, editors, *Genetic Programming*, pages 19–34. Cham, 2018. Springer International Publishing.
- [4] F. Assunção, N. Lourenço, P. Machado, and B. Ribeiro. DENSER: deep evolutionary network structured representation. *Genetic Programming and Evolvable Machines*, 20(1):5–35, 2019.
- [5] T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Oxford, UK, 1996.
- [6] B. Baker, O. Gupta, N. Naik, and R. Raskar. Designing neural network architectures using reinforcement learning. *CoRR*, abs/1611.02167, 2016.
- [7] P. Baldi. Autoencoders, unsupervised learning, and deep architectures. In I. Guyon, G. Dror, V. Lemaire, G. Taylor, and D. Silver, editors, *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, pages 37–49. Bellevue, Washington, USA, 02 Jul 2012. PMLR.
- [8] H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck. *Route Planning in Transportation Networks*, pages 19–80. Springer International Publishing, Cham, 2016.
- [9] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *CoRR*, abs/1207.4708, 2012.
- [10] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press, 2007.
- [11] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305, 2012.

- [12] J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28, ICML13*, page I1151123. JMLR.org, 2013.
- [13] H.-G. Beyer and H.-P. Schwefel. Evolution strategies a comprehensive introduction. *Natural Computing: An International Journal*, 1(1):352, May 2002.
- [14] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. SMASH: one-shot model architecture search through hypernetworks. *CoRR*, abs/1708.05344, 2017.
- [15] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016. cite arxiv:1606.01540.
- [16] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang. Efficient architecture search by network transformation. In S. A. McIlraith and K. Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 2787–2794. AAAI Press, 2018.
- [17] T. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma. Pcanet: A simple deep learning baseline for image classification? *IEEE Transactions on Image Processing*, 24(12):5017–5032, 2015.
- [18] F. Charte, A. J. Rivera, F. Martinez, and M. J. del Jesus. Evoaaa: An evolutionary methodology for automated neural autoencoder architecture search. *Integrated Computer-Aided Engineering*, Pre-press(Pre-press):1–21, 2020.
- [19] S. Chen, G. Liu, C. Wu, Z. Jiang, and J. Chen. Image classification with stacked restricted boltzmann machines and evolutionary function array classification voter. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 4599–4606, 2016.
- [20] X. Chen and X. Lin. Big data deep learning: Challenges and perspectives. *IEEE Access*, 2(1):514–525, 2014.
- [21] C. A. C. Coello. Evolutionary multi-objective optimization: a historical view of the field. *IEEE Computational Intelligence Magazine*, 1(1):28–36, Feb 2006.
- [22] C. A. Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1(3):269–308, 1999.
- [23] A. Darwish, A. E. Hassanien, and S. Das. A survey of swarm and evolutionary computing approaches for deep learning. *Artificial Intelligence Review*, pages 1–46, 2019.
- [24] O. E. David and I. Greental. Genetic algorithms for evolving deep neural networks. In *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation, GECCO Comp 14*, page 14511452, New York, NY, USA, 2014. Association for Computing Machinery.
- [25] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 2001.
- [26] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2002.
- [27] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.
- [28] T. Desell. Large scale evolution of convolutional neural networks using volunteer computing. In P. A. N. Bosman, editor, *Genetic and Evolutionary Computation Conference, Berlin, Germany, July 15-19, 2017, Companion Material Proceedings*, pages 127–128. ACM, 2017.
- [29] G. Desjardins and Y. Bengio. Empirical evaluation of convolutional rbms for vision. Technical Report 1327, Département d’Informatique et de Recherche Opérationnelle, Université de Montréal, 2008.
- [30] E. Dufourq and B. A. Bassett. Eden: Evolutionary deep networks for efficient machine learning. In *2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech)*, pages 110–115, 2017.
- [31] A. E. Eiben and J. Smith. From evolutionary computation to the evolution of things. *Nature*, 521:476–482, 28 May 2015.
- [32] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer Verlag, 2003.
- [33] T. Elsken, J.-H. Metzen, and F. Hutter. Simple and efficient architecture search for convolutional neural networks, 2017.
- [34] T. Elsken, J. H. Metzen, and F. Hutter. *Neural Architecture Search*, pages 63–77. Springer International Publishing, Cham, 2019.
- [35] C. Fernando, D. Banarse, M. Reynolds, F. Besse, D. Pfau, M. Jaderberg, M. Lanctot, and D. Wierstra. Convolution by evolution: Differentiable pattern producing networks. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO 16*, page 109116, New York, NY, USA, 2016. Association for Computing Machinery.
- [36] A. Fischer and C. Igel. An introduction to restricted boltzmann machines. In L. Alvarez, M. Mejail, L. Gomez, and J. Jacobo, editors, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 14–36, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- [37] P. Fleming and R. Purshouse. Evolutionary algorithms in control systems engineering: a survey. *Control Engineering Practice*, 10(11):1223 – 1241, 2002.
- [38] D. Floreano, P. Dürr, and C. Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, 2008.
- [39] L. Fogel, A. Owens, and M. Walsh. *Artificial intelligence through simulated evolution*. Wiley, Chichester, WS, UK, 1966.
- [40] E. Galván-López. *An analysis of the effects of neutrality on problem hardness for evolutionary algorithms*. PhD thesis, University of Essex, Colchester, UK, 2009.
- [41] E. Galván-López, S. Dignum, and R. Poli. The effects of constant neutrality on performance and problem hardness in GP. In M. O’Neill, L. Vanneschi, S. M. Gustafson, A. Esparcia-Alcázar, I. D. Falco, A. D. Cioppa, and E. Tarantino, editors, *Genetic Programming, 11th European Conference, EuroGP 2008, Naples, Italy, March 26-28, 2008. Proceedings*, volume 4971 of *Lecture Notes in Computer Science*, pages 312–324. Springer, 2008.
- [42] E. Galván-López and R. Poli. An empirical investigation of how and why neutrality affects evolutionary search. In M. Cattolico, editor, *Genetic and Evolutionary Computation Conference, GECCO 2006, Proceedings, Seattle, Washington, USA, July 8-12, 2006*, pages 1149–1156. ACM, 2006.
- [43] E. Galván-López and R. Poli. Some steps towards understanding how neutrality affects evolutionary search. In T. P. Runarsson, H. Beyer, E. K. Burke, J. J. Merelo Guervós, L. D. Whitley, and X. Yao, editors, *Parallel Problem Solving from Nature - PPSN IX, 9th International Conference, Reykjavik, Iceland, September 9-13, 2006, Proceedings*, volume 4193, pages 778–787. Springer, 2006.
- [44] E. Galván-López and R. Poli. An empirical investigation of how degree neutrality affects GP search. In A. H. Aguirre, R. M. Borja, and C. A. R. García, editors, *MICAI 2009: Advances in Artificial Intelligence, 8th Mexican International Conference on Artificial Intelligence, Guanajuato, Mexico, November 9-13, 2009. Proceedings*, volume 5845 of *Lecture Notes in Computer Science*, pages 728–739. Springer, 2009.
- [45] E. Galván-López, R. Poli, A. Kattan, M. O’Neill, and A. Brabazon. Neutrality in evolutionary algorithms... what do we know? *Evolving Systems*, 2(3):145–163, 2011.
- [46] F. Gers. Learning to forget: continual prediction with lstm. *IET Conference Proceedings*, pages 850–855(5), January 1999.
- [47] F. A. Gers and E. Schmidhuber. Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340, 2001.
- [48] C. Goerick and T. Rodemann. Evolution strategies: An alternative to gradient based learning.
- [49] D. E. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley, 1989.
- [50] I. Gonçalves, M. Seca, and M. Castelli. *Explorations of the Semantic Learning Machine Neuroevolution Algorithm: Dynamic Training Data Use, Ensemble Construction Methods, and Deep Learning Perspectives*, pages 39–62. Springer International Publishing, Cham, 2020.
- [51] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.
- [52] A. Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013.
- [53] K. Greff, R. K. Srivastava, J. Koutnk, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2017.
- [54] J. Hajewski, S. Oliveira, and X. Xing. Distributed evolution of deep autoencoders, 2020.
- [55] N. Hansen, S. D. Miller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary Computation*, 11(1):1–18, 2003.
- [56] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation.

- In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 312–317, 1996.
- [57] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2):159195, June 2001.
- [58] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.
- [59] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):15271554, July 2006.
- [60] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.
- [61] C. Hong, J. Yu, J. Wan, D. Tao, and M. Wang. Multimodal deep autoencoder for human pose recovery. *IEEE Transactions on Image Processing*, 24(12):5659–5670, 2015.
- [62] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.
- [63] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex. *The Journal of Physiology*, 160(1):106–154, 1962.
- [64] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In F. R. Bach and D. M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015.
- [65] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision*, pages 2146–2153, 2009.
- [66] Y. Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61–70, 2011.
- [67] R. Józefowicz, W. Zaremba, and I. Sutskever. An empirical exploration of recurrent network architectures. In F. R. Bach and D. M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2342–2350. JMLR.org, 2015.
- [68] K. Kandasamy, W. Neiswanger, J. Schneider, B. Póczos, and E. P. Xing. Neural architecture search with bayesian optimisation and optimal transport. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 2020–2029, 2018.
- [69] S. Khadka, J. J. Chung, and K. Tumer. Neuroevolution of a modular memory-augmented neural network for deep memory problems. *Evol. Comput.*, 27(4):639–664, 2019.
- [70] S. Khadka and K. Tumer. Evolution-guided policy gradient in reinforcement learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS18*, page 11961208, Red Hook, NY, USA, 2018. Curran Associates Inc.
- [71] Y.-H. Kim, B. Reddy, S. Yun, and C. Seo. Nemo : Neuro-evolution with multiobjective optimization of deep neural network for speed and accuracy. 2017.
- [72] M. Kimura. Evolutionary rate at the molecular level. *Nature*, 217:624–626, 1968.
- [73] M. Kimura. *The Neutral Theory of Molecular Evolution*. Cambridge University Press, 1983.
- [74] H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4, 1990.
- [75] J. Koza, M. Keane, M. Streeter, W. Mydlowec, J. Yu, and G. Lanza. Genetic programming iv: Routine human-competitive machine intelligence. 01 2003.
- [76] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.
- [77] J. R. Koza. Human-competitive results produced by genetic programming. *Genetic Programming and Evolvable Machines*, 11(3/4):251–284, Sept. 2010. Tenth Anniversary Issue: Progress in Genetic Programming and Evolvable Machines.
- [78] A. Krizhevsky, V. Nair, and G. Hinton. Cifar-10 (canadian institute for advanced research).
- [79] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):8490, May 2017.
- [80] S. Lander and Y. Shang. Evoae – a new evolutionary method for training autoencoders for deep learning networks. In *2015 IEEE 39th Annual Computer Software and Applications Conference*, volume 2, pages 790–795, 2015.
- [81] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th International Conference on Machine Learning, ICML 07*, page 473480, New York, NY, USA, 2007. Association for Computing Machinery.
- [82] H. Larochelle, M. Mandel, R. Pascanu, and Y. Bengio. Learning algorithms for the classification restricted boltzmann machine. *Journal of Machine Learning Research*, 13(22):643–669, 2012.
- [83] Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [84] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.
- [85] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [86] J. Lehman and K. Stanley. *Novelty Search and the Problem with Objectives*, pages 37–56. 11 2011.
- [87] J. Lehman and K. O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evol. Comput.*, 19(2):189223, June 2011.
- [88] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman. Random feedback weights support learning in deep neural networks. *arXiv preprint arXiv:1411.0247*, 2014.
- [89] M. Lindauer and F. Hutter. Best practices for scientific research on neural architecture search, 2019.
- [90] C. Liu, B. Zoph, J. Shlens, W. Hua, L. Li, L. Fei-Fei, A. L. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. *CoRR*, abs/1712.00559, 2017.
- [91] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu. Hierarchical representations for efficient architecture search. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.
- [92] J. Liu, M. Gong, Q. Miao, X. Wang, and H. Li. Structure learning for deep neural networks based on multiobjective optimization. *IEEE Transactions on Neural Networks and Learning Systems*, 29(6):2450–2463, 2018.
- [93] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.
- [94] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf. Nsga-net: Neural architecture search using multi-objective genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 19*, page 419427, New York, NY, USA, 2019. Association for Computing Machinery.
- [95] X. Luo, X. Li, Z. Wang, and J. Liang. Discriminant autoencoder for feature extraction in fault diagnosis. *Chemometrics and Intelligent Laboratory Systems*, 192:103814, 2019.
- [96] M. Mandischer. A comparison of evolution strategies and backpropagation for neural network training. *Neurocomputing*, 42(1):87–117, 2002. Evolutionary neural systems.
- [97] A. Martn, R. Lara-Cabrera, F. Fuentes-Hurtado, V. Naranjo, and D. Camacho. Evodeep: A new evolutionary approach for automatic deep neural networks parametrisation. *Journal of Parallel and Distributed Computing*, 117:180–191, 2018.
- [98] Z. Michalewicz. *How to Solve It: Modern Heuristics 2e*. Springer-Verlag, Berlin, Heidelberg, 2010.
- [99] R. Miikkulainen, J. Z. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat. Evolving deep neural networks. *CoRR*, abs/1703.00548, 2017.
- [100] J. F. Miller. *Cartesian Genetic Programming*, pages 17–34. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
- [101] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.
- [102] A. Mohamed, T. N. Sainath, G. Dahl, B. Ramabhadran, G. E. Hinton, and M. A. Picheny. Deep belief networks using discriminative features for phone recognition. In *2011 IEEE International Conference on*

- Acoustics, Speech and Signal Processing (ICASSP)*, pages 5060–5063, 2011.
- [103] D. J. Montana and L. Davis. Training feedforward neural networks using genetic algorithms. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 1, IJCAI89*, page 762767, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.
- [104] G. Morse and K. O. Stanley. Simple evolutionary optimization can rival stochastic gradient descent in neural networks. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO 16*, page 477484, New York, NY, USA, 2016. Association for Computing Machinery.
- [105] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML10*, page 807814, Madison, WI, USA, 2010. Omnipress.
- [106] R. Negrinho and G. Gordon. Deeparchitect: Automatically designing and training deep architectures, 2017.
- [107] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. *NIPS*, 01 2011.
- [108] A. Ororbia, A. ElSaid, and T. Desell. Investigating recurrent neural network memory structures using neuro-evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 19*, page 446455, New York, NY, USA, 2019. Association for Computing Machinery.
- [109] K. Pawełczyk, M. Kawulok, and J. Nalepa. Genetically-trained deep neural networks. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 18*, page 6364, New York, NY, USA, 2018. Association for Computing Machinery.
- [110] L. Peng, S. Liu, R. Liu, and L. Wang. Effective long short-term memory with differential evolution algorithm for electricity price prediction. *Energy*, 162:1301 – 1314, 2018.
- [111] R. Poli and E. Galván-López. On the effects of bit-wise neutrality on fitness distance correlation, phenotypic mutation rates and problem hardness. In C. R. Stephens, M. Toussaint, D. Whitley, and P. F. Stadler, editors, *Foundations of Genetic Algorithms*, pages 138–164, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
- [112] R. Poli and E. Galván-López. The effects of constant and bit-wise neutrality on problem hardness, fitness distance correlation and phenotypic mutation rates. *IEEE Trans. Evolutionary Computation*, 16(2):279–300, 2012.
- [113] V. W. Porto, D. B. Fogel, and L. J. Fogel. Alternative neural network training methods. *IEEE Expert: Intelligent Systems and Their Applications*, 10(3):1622, June 1995.
- [114] K. Price, R. Storn, and J. Lampinen. *Differential Evolution-A Practical Approach to Global Optimization*, volume 141. 01 2005.
- [115] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama. Opposition-based differential evolution. *IEEE Transactions on Evolutionary Computation*, 12(1):64–79, 2008.
- [116] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. Regularized evolution for image classifier architecture search. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 4780–4789. AAAI Press, 2019.
- [117] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML17*, page 29022911. JMLR.org, 2017.
- [118] I. Rechenberg. Evolutionsstrategien. In B. Schneider and U. Ranft, editors, *Simulationmethoden in der Medizin und Biologie*, pages 83–114, Berlin, Heidelberg, 1978. Springer Berlin Heidelberg.
- [119] I. Rechenberg. Evolution strategy: Nature’s way of optimization. In H. W. Bergmann, editor, *Optimization: Methods and Applications, Possibilities and Limitations*, pages 106–126, Berlin, Heidelberg, 1989. Springer Berlin Heidelberg.
- [120] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning Internal Representations by Error Propagation*, page 318362. MIT Press, Cambridge, MA, USA, 1986.
- [121] C. Ryan, J. Collins, and M. O. Neill. Grammatical evolution: Evolving programs for an arbitrary language. In W. Banzhaf, R. Poli, M. Schoenauer, and T. C. Fogarty, editors, *Genetic Programming*, pages 83–96, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.
- [122] R. Salakhutdinov and G. Hinton. Deep boltzmann machines. In D. van Dyk and M. Welling, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 448–455, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.
- [123] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever. Evolution strategies as a scalable alternative to reinforcement learning, 2017.
- [124] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117, 2015.
- [125] H.-P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, Inc., USA, 1981.
- [126] F. Sehnke, C. Osendorfer, T. Rckstie, A. Graves, J. Peters, and J. Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551 – 559, 2010. The 18th International Conference on Artificial Neural Networks, ICANN 2008.
- [127] T. Shinozaki and S. Watanabe. Structure discovery of deep neural network based on evolutionary algorithms. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4979–4983, 2015.
- [128] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, Jan. 2016.
- [129] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
- [130] D. R. So, Q. V. Le, and C. Liang. The evolved transformer. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5877–5886. PMLR, 2019.
- [131] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323 – 332, 2012. Selected Papers from IJCNN 2011.
- [132] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen. Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1:24–35, 2019.
- [133] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evol. Comput.*, 10(2):99127, June 2002.
- [134] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *ArXiv*, abs/1712.06567, 2017.
- [135] M. Suganuma, M. Ozay, and T. Okatani. Exploiting the potential of standard convolutional autoencoders for image restoration by evolutionary search. In J. G. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4778–4787. PMLR, 2018.
- [136] M. Suganuma, S. Shirakawa, and T. Nagao. A genetic programming approach to designing convolutional neural network architectures. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5369–5373. International Joint Conferences on Artificial Intelligence Organization, 7 2018.
- [137] Y. Sun, H. Wang, B. Xue, Y. Jin, G. G. Yen, and M. Zhang. Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor. *IEEE Transactions on Evolutionary Computation*, 24(2):350–364, 2020.
- [138] Y. Sun, X. Wang, and X. Tang. Hybrid deep learning for face verification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):1997–2009, 2016.
- [139] Y. Sun, B. Xue, M. Zhang, and G. G. Yen. Completely automated cnn architecture design based on blocks. *IEEE Transactions on Neural Networks and Learning Systems*, 31(4):1242–1254, 2020.
- [140] Y. Sun, B. Xue, M. Zhang, and G. G. Yen. Evolving deep convolutional neural networks for image classification. *IEEE Transactions on Evolutionary Computation*, 24(2):394–407, 2020.
- [141] Y. Sun, G. G. Yen, and Z. Yi. Evolving unsupervised deep neural networks for learning meaningful representations. *IEEE Trans. Evolutionary Computation*, 23(1):89–103, 2019.
- [142] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

- [143] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1–9. IEEE Computer Society, 2015.
- [144] G. J. van Wyk and A. S. Bosman. Evolutionary neural architecture search for image restoration. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019.
- [145] B. Wang, Y. Sun, B. Xue, and M. Zhang. A hybrid differential evolution approach to designing deep convolutional neural networks for image classification. In T. Mitrovic, B. Xue, and X. Li, editors, *AI 2018: Advances in Artificial Intelligence - 31st Australasian Joint Conference, Wellington, New Zealand, December 11-14, 2018, Proceedings*, volume 11320 of *Lecture Notes in Computer Science*, pages 237–250. Springer, 2018.
- [146] S. Wright. The role of mutation, inbreeding, crossbreeding and selection in evolution. In *Proceedings of the Sixth International Congress on Genetics*, volume 1, page 356366, 1932.
- [147] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.
- [148] L. Xie and A. Yuille. Genetic cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1388–1397, 2017.
- [149] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.
- [150] S. Zagoruyko and N. Komodakis. Wide residual networks. In R. C. Wilson, E. R. Hancock, and W. A. P. Smith, editors, *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. BMVA Press, 2016.
- [151] C. Zhang, P. Lim, A. K. Qin, and K. C. Tan. Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2306–2318, 2017.
- [152] C. Zhang, K. C. Tan, H. Li, and G. S. Hong. A cost-sensitive deep belief network for imbalanced classification. *IEEE Trans. Neural Networks Learn. Syst.*, 30(1):109–122, 2019.
- [153] Q. Zhao, D. Zhang, and H. Lu. A direct evolutionary feature extraction algorithm for classifying high dimensional data. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1, AAAI06*, page 561566. AAAI Press, 2006.
- [154] Z. Zhong, J. Yan, and C. Liu. Practical network blocks design with q-learning. *CoRR*, abs/1708.05552, 2017.
- [155] H. Zhu, M. Akrouf, B. Zheng, A. Pelegris, A. Jayarajan, A. Phanishayee, B. Schroeder, and G. Pekhimenko. Benchmarking and analyzing deep neural network training. In *2018 IEEE International Symposium on Workload Characterization (IISWC)*, pages 88–100, 2018.
- [156] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016.
- [157] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2018.