ARCHIVIO ISTITUZIONALE
DELLA RICERCA

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

Shallow2Deep: Restraining Neural Networks Opacity through Neural Architecture Search

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:
Andrea Agiollo, G.C. (2021). Shallow2Deep: Restraining Neural Networks Opacity through Neural Architecture Search. Cham : Springer [10.1007/978-3-030-82017-6_5].

Availability:
This version is available at: https://hdl.handle.net/11585/838540 since: 2021-11-15

Published:

DOI: http://doi.org/10.1007/978-3-030-82017-6_5

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/).
When citing, please refer to the published version.

(Article begins on next page)

07 November 2026

This is the final peer-reviewed accepted manuscript of:

Agiollo, A., Ciatto, G., Omicini, A. (2021). *Shallow2Deep*: Restraining Neural Networks Opacity Through Neural Architecture Search. In: Calvaresi, D., Najjar, A., Winikoff, M., Främling, K. (eds) Explainable and Transparent AI and Multi-Agent Systems. EXTRAAMAS 2021. Lecture Notes in Computer Science(), vol 12688. Springer, Cham.

The final published version is available online at: **https://doi.org/10.1007/978-3-030-82017-6_5**

Rights / License:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/)

When citing, please refer to the published version.

Shallow2Deep: Restraining Neural Networks Opacity through Neural Architecture Search

Andrea Agiollo[✉1,2 0000−0003−0531−1978], Giovanni Ciatto[1 0000−0002−1841−8996], and Andrea Omicini[1 0000−0002−6655−3869]

[1] Alma Mater Studiorum—University of Bologna, Italy
{andrea.agiollo, giovanni.ciatto, andrea.omicini}@unibo.it
[2] The Research Hub by Electrolux Professional S.p.A., 33170 Pordenone (PN), Italy
andrea.agiollo@electroluxprofessional.com

Abstract. Recently, the Deep Learning (DL) research community has focused on developing efficient and highly performing Neural Networks (NN). Meanwhile, the eXplainable AI (XAI) research community has focused on making Machine Learning (ML) and Deep Learning methods *interpretable* and *transparent*, seeking *explainability*. This work is a preliminary study on the applicability of Neural Architecture Search (NAS) (a sub-field of DL looking for automatic design of NN structures) in XAI. We propose *Shallow2Deep*, an evolutionary NAS algorithm that exploits local variability to restrain opacity of DL-systems through NN architectures simplification. *Shallow2Deep* effectively reduces NN complexity – therefore their opacity – while reaching state-of-the-art performances. Unlike its competitors, *Shallow2Deep* promotes variability of localised structures in NN, helping to reduce NN opacity. The proposed work analyses the role of local variability in NN architectures design, presenting experimental results that show how this feature is actually desirable.

Keywords: Neural Architecture Search · Evolutionary Algorithm · Opacity · Interpretability

1 Introduction

Data-driven intelligent systems pervade modern society. The recent advancements of Machine Learning (ML) and Deep Learning (DL) are boosting the adoption of neural networks (NN) in several contexts, including, but not limited to, healthcare, finance, law, and domestic appliances. For this reason, the exploitation of DL-enabled systems in industrial applications and every-day life requires precision and efficiency. However, both features come at a price. In fact, state-of-the-art neural architectures are characterised by an ever-increasing *structural* complexity – in terms of layers, neurons, and their connections –, which is expected to make neural networks even more precise and accurate.

The structural complexity of NN, however, brings about a number of drawbacks. For instance, it makes training more eager for computational resources and data. Furthermore, it represents a bottleneck in the engineering process of

DL systems—which is commonly performed by data scientists, manually. Finally, and more importantly, it contributes to the well-known *opacity* issues making NN inner operation hard to understand for human beings—there including both expert practitioners and users. For all these reasons, research efforts devoted to the identification of *small* – i.e. structurally simpler – yet highly-performing neural architectures are gaining momentum within the DL community [17,39].

Opacity of DL systems, in particular, is a critical aspect which should be reduced and possibly avoided. As suggested by the eXplainable Artificial Intelligence (XAI) initiative [11,1], there is an urgent need for making the operation and outcomes of modern intelligent systems more human-interpretable. So far, several means have been proposed into the XAI literature to serve these purposes, following as many strategies. When it comes to DL, however, most existing techniques focus on either *(i)* easing the *inspection* of NN – via visualisation facilities –, or *(ii)* enabling their replacement with transparent models of similar performance—such as rules lists or decision trees. In other words, not enough care seems to be given to the problem of making DL models more transparent.

Arguably, a possible way to decrease NN opacity is to reduce their structural complexity, while preserving their predictive performance. Many works focusing on NN explainability can produce rules lists or decision trees equivalent models for small NN [8,40]: e.g., RefAnn [36] is a rule extraction procedure tailored on neural networks having a *single* hidden layer. While such approaches can hardly be applied to complex NN, we argue that an automated procedure capable of reducing the internal structure of a NN may pave the way towards a wider adoption of algorithms that would otherwise be inapplicable.

Generally speaking, complexity reduction may bring benefits at several levels, including transparency and training time, other than the capability to extract human-readable rules or trees out of NN. However, a limiting factor along this line is that, currently, NN structures are handcrafted by human experts via a trial-and-error procedure, targeting predictive performance rather than lower structural complexity. Furthermore, experts' experience and intuition play a pivotal role in the process, making it hard to automate and reproduce. This is where Neural Architecture Search (NAS) [12,42] comes into play. The general goal of NAS is to *automate* the identification of the best NN structure for a given task. Several approaches are being explored, modelling the NAS as a search problem in the space of all possible NN architectures. To the best of our knowledge, however, no work so far focuses on controlling network structural complexity.

Accordingly, in this paper we propose *Shallow2Deep*, a novel NAS algorithm aimed at keeping NN structural complexity under control. Our algorithm allows data scientists to *automatically* and efficiently detect highly-predictive architectures for convolutional NN targeting pattern matching tasks—such as image or speech recognition. It enforces structural constraints over the searched NN architecture, limiting the NN structural complexity and therefore its opacity. In other words, *Shallow2Deep* fits NN design by providing a means to control the depth of a NN, possibly enabling, e.g., the applicability of rule-extraction algorithms in complex tasks.

Our solution differs from other NAS approaches in a number of ways. First, it promotes local *variability* in NN architectures—meaning that it supports and encourages variability in the different layers composing a NN. Then, it favours local *specialisation* of NN sub-structures—thus letting each layer of the NN specialise on different tasks, depending on their depth. Finally, it promotes progressive complexity, avoiding *overthinking*—a well-acknowledged [19] tendency of deep NN to learn too many concepts, becoming more complex than needed. We present a full operational formalisation of the *Shallow2Deep* algorithm along with a number of experiments showing its practical feasibility and versatility.

2 Background

2.1 Neural Architecture Search

Neural networks are biologically-inspired computational models, made of several elementary units (neurons) interconnected into a *directed-acyclic* graph (DAG) via *weighted* synapses. NN can be *trained* on data via backpropagation [16] and exploited into both supervised and unsupervised learning tasks such as classification, regression, and anomaly detection. The training phase makes a NN learn from data. Yet, only network synapses weights are modified in this phase, whereas its overall graph structure (topology henceforth) is not allowed to vary. It is rather assumed to be *manually* engineered by data scientists.

Convolutional Neural Networks (CNN) are particular sorts of NN whose topology consists of a cascade of *convolutional* layers. In other words, CNN can learn how to apply a number of convolution operations [30] to the data. Convolutions let the network spot relevant features into the input data, at possibly different scales. Thus, CNN are primarily used to solve complex pattern-recognition tasks, such as in computer vision or speech recognition. Yet, how many convolutions a network may learn as well as their interconnections depend on NN topology. Again, this implementation detail requires human intervention.

Traditionally, NN development workflows are deeply influenced by the choices by human experts. Network architectures represent the most relevant aspect requiring human contribution. However, as neither theorems nor methods ensure optimal results, human choices may lead to sub-optimal or inefficient solutions.

To avoid inefficiencies introduced by human errors in NN design, *neural architecture search* (NAS) has been proposed [26]. NAS automates network architecture engineering: it aims at learning a network topology that can achieve reasonably-good performances on specific tasks, by letting a search algorithm look for the best network topology among the admissible ones.

To keep the computational complexity of NAS acceptable, several approaches have been proposed in the literature. Virtually all of them try to *(i)* reduce the search space size and *(ii)* control the whole search duration by leveraging on a *greedy* or *evolutionary* search strategy.

A common means to restrict the search space is to assume the network topology to be composed by a sequence of units called *cells*. Each cell contains a number of *blocks*, connected over a DAG structure. Blocks, in turn, are groups of

neurons having a predefined internal organisation—commonly corresponding to a particular mathematical operator. In CNN, for instance, blocks are commonly constrained to represent convolutional layers, each one representing different sorts of convolutional filters—e.g. 3×3, 5×5, etc. Directed connections among any two cells A and B are modelled as directed connections among the output block of A and the input block of B.

Within the scope of this paper, we denote by $\mathbb{O}$ the *operation set*, i.e. the set of all possible *sorts* of blocks for any given NAS problem. We assume $\mathbb{O}$ is always a finite-cardinality set. For instance, in the particular case of CNN, $\mathbb{O}$ may contain different sorts of convolutional layers (e.g. $\mathbb{O} = \{3 \times 3, 5 \times 5, 7 \times 7\}$).

State-of-the-art NAS approaches mostly differ in which and how many blocks and cells are exploited, how these can be connected with each others, or which (meta-)heuristic the search of the optimal topology leverages upon. For instance, a method is proposed in [22] where NN is built as a sequence of identical cells—so that only the internal structure of one cell is optimised. In [32] a *regularised* evolutionary meta-heuristic is presented introducing an age property to favour younger genotypes. The approach aims at optimising two sorts of cells – "normal" or "reduction" cells – which are concatenated in a predefined way to obtain the final NN. Similarly, a *continuous* evolutionary approach is proposed in [45] sampling the population of different generations from a super network $\mathcal{N}$ of shared parameters to search for normal and reduction cells. In [7], a probabilistic approach is applied to reduce the search memory requirements to obtain the best structures for the normal and reduction cell.

Another common approach is to fix cells operations and look for the best cell combination that can compose the NN. In [9] a multi-objective oriented algorithm is presented exploiting both evolution and reinforcement learning to search for the best composition of various predefined cells. In [38] a method is presented that explicitly incorporates model complexity into the objective function while searching for the best cell sequence. However, cells are selected from a predefined pool, and they are not optimised any further. Conversely, a differentiable NAS framework is proposed in [43], searching for the best cell placement in the NN structure. Finally, in [5] a method is presented that can search NN architectures avoiding proxies and limitations typical of other approaches—e.g., training on a smaller dataset, learning with only few blocks, training for few epochs.

Reduction in NN complexity can also be achieved via network pruning techniques [24]. However, here we focus on NAS techniques only, since network pruning is a post-hoc technique not taking into account NN structure, as it is applied *after* training—when the structure has already been fixed. For this reason, structural inefficiencies of NN can not be tackled using network pruning techniques.

To the best of our knowledge, our work represents the first attempt to produce a NAS strategy capable of identifying both a good cell structure and a global architecture, avoiding constraints on architecture design. In other words, our approach improves the NN performance over complexity ratio, allowing *Shallow2Deep* to produce smaller – yet reasonably performing – NN architectures.

2.2 eXplainable AI *vs.* Neural Networks

Neural networks and computer vision are commonly exploited behind the scenes of intelligent systems involved in several critical applications (e.g., intelligent medicine, automotive, etc.). There, NN provide intelligent systems with high-precision pattern recognition capabilities. However, as for most ML methods, NN engineering only focuses on attaining high predictive performance, whereas poor care is given when it comes to determine *why* NN provide a particular outcome. To complicate this issue, advanced applications – such as computer vision – may easily involve *deep* NN, having an intricate internal structure which makes them hard to analyse and understand—even for experts. Such intricacy is described into the literature as "opacity" [21]—a feature characterising most ML algorithms up to some extent, which are also known as "black boxes" for that very same reason [14].

Recently, the opacity issue characterising ML and, in particular, NN reached the general public attention—also because of the GDPR regulations[3]. Accordingly, safety- or privacy-critical intelligent applications leveraging on ML should be designed accounting for properties such as interpretability, reliability, and transparency [41]. In other words, human users must be able to understand the criteria behind machine-aided or -driven decisions.

The XAI community [3] is currently intercepting those needs by proposing methods aimed at tackling the opacity issues that characterise ML- and DL-powered systems. Most methods proposed so far essentially focus on *(i)* inspection or visualisation [48,4] techniques, aimed at "debugging" the inner functioning of NN; *(ii)* heatmaps [2,35], or feature relevance analyses [28,18], aimed at analysing a network behaviour w.r.t. its inputs; or *(iii)* symbolic knowledge (e.g., rules, or trees) extraction algorithms [49,6], aimed at distilling human-intelligible information out of the intricate structure of sub-symbolic predictors. In other words, following a nomenclature introduced in [10], current methods either attempt to make NN more easily interpretable or *a-posteriori* explainable.

NN opacity is deeply entangled with their structural complexity. The more NN architectures are intricate the less NN are interpretable. By keeping NN structural complexity under control, data scientists may limit their opacity. In turn, limiting NN opacity, plays a fundamental role in XAI as it may make NN both *a-priori* interpretable and *a-posteriori* explainable. Accordingly, to the best of our knowledge, our work represents the first attempt to control NN opacity – i.e. structural complexity – while automating the search for a well-performing architecture.

3 Shallow2Deep

In this section we present *Shallow2Deep*, a novel exhaustive NAS algorithm: we first present its architecture design along with the corresponding search space (3.1), then we discuss its overall working principle, its fundamental hypotheses, and the details of its composing modules.

[3] `https://eur-lex.europa.eu/eli/reg/2016/679/oj`

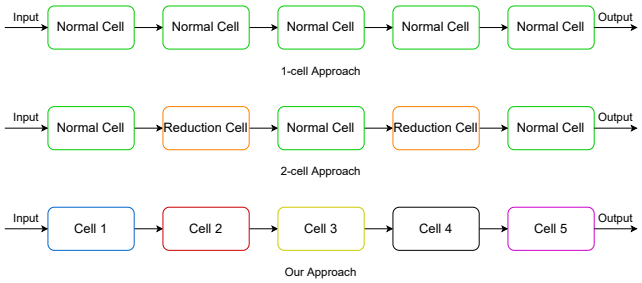

Fig. 1: *Shallow2Deep* avoids architecture design limitations, common in other NAS algorihtms. Here $C = 5$.

3.1 Architecture Design

Most popular NAS approaches blindly build a NN architecture by repeating the same *elementary* structure (cell) several times—assuming that the internal structure of a cell has been manually optimised. This sounds like a reasonable approach, considering the history of NN development and validation. Handcrafted successful networks (e.g., VGG [37], ResNet [15], Mobilenet [17], etc.) are composed by repetitions of a certain peculiar element (e.g., convolutions, skip connections, inverted mobile bottleneck, etc.). Furthermore, repetitions lead to a reduction of the search space size, when an effective combination of the elementary cells must be automatically computed.

However, although understandable from a computational perspective, such an approach is not reasonable in terms of predictive capability. Relying on the same elementary structure at different depth levels of the network architecture may hinder the predictive performance of the resulting NN as a whole. In fact, assuming a particular elementary structure is good enough to let a network's *shallow*[4]layers perform valuable feature extraction, it is unlikely that the same structure is equally good to provide that network's *deep* layers with more sophisticated pattern matching capabilities. The different layers of a well-trained NN are expected to perform totally-different feature-mining tasks. This is why we argue that the best elementary structure for shallow and deep layers of NN are not architecturally equal. *Shallow2Deep* builds on this, providing a means to look for a good cell structure specification for both shallow and deep layers.

More precisely, *Shallow2Deep* constructs NN classifiers with a fixed number C of *different* cells. Cells can differ in terms of the topology they assume and the operations they apply. We define the difference between cells w.r.t. topology and operations as *structure variability*. *Shallow2Deep* promotes local *structure variability* in NN architectures, avoiding design limitations.

Figure 1 highlights the main difference among state-of-the-art NAS mechanisms and our approach: *Shallow2Deep* lets each cell vary independently from

[4] By "shallow" (resp. "deep") layers of a NN we mean the *inner* layers close to the input (resp. output) neurons.

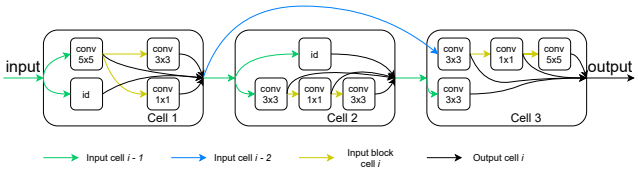

Fig. 2: Blocks in *Shallow2Deep* cells can get input from previous cell (green), the cell before that (blue) or any other block in the cell (yellow). Cell output is obtained concatenating outputs of all blocks belonging to the cell (black). Here $C = 3$ and $B = 4$.

each other. However, similarly to other authors, we assume cells to be ordered from shallow side to the deep side. Accordingly, the 1^{st} cell is the closest one to the inputs, while the last one is the closest to the outputs.

In particular, *Shallow2Deep* lets each cell contain B blocks—being B a positive and finite integer. Each block represents a particular sort of NN layer. Following a convention introduced in [46], we denote by $\mathbb{O}$ the *finite* set of all possible sorts of blocks—which in turn depend on the particular task the target NN aims to solve. For instance, if the considered NN targets image recognition tasks, we let $\mathbb{O}$ contain simple convolutions, other than the identity block—e.g. some $n \times n$ convolutional layers (for $n = 1, 3, 5, \ldots$), plus the identity layer $f(x) = x$.

Each block of the i^{th} cell can accept as input the output of $i-1^{th}$ and $i-2^{th}$ cells and of any other block in the same i^{th} cell. However, loops and cycles among blocks connections are not allowed. In other words, the blocks topology must be a DAG. This is necessary to preserve the feed-forward architecture of the NN. Furthermore, each block can provide output to any amount of other blocks.

The whole output of a cell is attained by concatenating the outputs of all blocks belonging to that cell, as in [22]. Figure 2 provides an example of an admissible topology that can be created by blocks and cells following the aforementioned rules.

3.2 Search Algorithm

Virtually all NAS algorithms proposed into the literature so far deal with reduced search spaces attained via strict architectural constraints. Conversely, our approach avoids the excessive simplification of the architectural design by allowing the internal structure of each cell to vary. A greedy search algorithm is then employed to automate the selection of the actual cells structures, in an iterative way. It relies on the successive search of locally-optimal cell structures proceeding from the shallower cells to the deeper ones.

As exemplified in fig. 3, *Shallow2Deep* consists of the iterative repetition of a local search algorithm aimed at selecting the (locally) best internal structure of the i^{th} cell. The search algorithm is repeated for all $i = 1, \ldots, C$, in such a way that the internal structure of the i^{th} cell is only optimised *after* that $(i-1)^{th}$ one

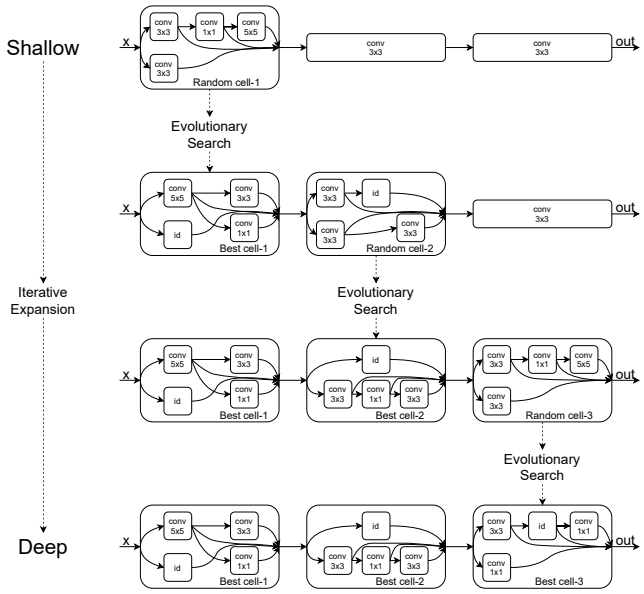

Fig. 3: *Shallow2Deep* iteratively searches for the best structure of cells going from shallow to deep ones. Shallow bests are searched using simple superstructures to increase the feature expressiveness and reduce training time. Local bests are kept fixed while deeper best are searched, reducing the complexity.

has already been optimised. In particular, here rely on an evolutionary algorithm to tackle local search. During local search, a population of NN is considered based on the structures that need to be analysed. The NN under examination are trained on a subset of the training set in order to find well behaving local structures—i.e. cell. To keep the whole process time-efficient, while optimising the i^{th} cell, all the j^{th} cells ($j \in \{1, \ldots, i-1\}$) are left unaffected by the training process. Moreover, to maximize the knowledge extracted at the i^{th} cell during its discovery process, all k^{th} cells ($k \in \{i+1, \ldots, C\}$) are built as bare as possible. Following literature, we consider bare cells to be composed of a single block applying a 3×3 convolution operation [13,37]. In other words, *Shallow2Deep* greedly proceeds from the shallowest cell to the deepest one.

While further details concerning our design choices are provided in section 4, some insights can be provided by the way a well-trained NN operates. The shallow layers of a NN aim to mine low level features. Complex features are extracted by deeper layers, reliably building on top of low level information. Therefore, *Shallow2Deep* searches for structures of deeper cells iteratively, building on the knowledge acquired at previous search steps.

Cell search The *Shallow2Deep* algorithm relies on a local search of the best performing structure for each cell of the NN. The task can be accomplished

through a variety of different search algorithms, from reinforcement learning to evolutionary algorithms [50,45]. In *Shallow2Deep* we exploit *evolutionary* (a.k.a. genetic) heuristic algorithms.

Evolutionary algorithms are a family of population-based metaheuristic optimization algorithms inspired by biological evolution. They commonly rely on a set of predefined stochastic mechanisms – namely, generation, mutation, selection, mating, fitness, etc – which let the algorithm randomly explore a vast search space in a smart way. Technically, these algorithms attempt to solve an optimisation problem by generating population of random solutions for the problem at hand, and by simulating evolution for a predefined amount of iterations—a.k.a. *generations*. Solutions are more or less likely, to survive among generations depending on their *fitness*—i.e. a measure of the quality of a particular solution w.r.t. the problem at hand. To prevent the search to step into local optima, evolutionary algorithms may exploit a number of strategies to introduce more randomness in the precess, such as mutations—meaning that solutions may randomly mutate while stepping through generations.

We choose to rely on evolutionary algorithms because of their *(i)* flexibility, *(ii)* support to *space pruning* [25] – a feature that we plan to support in the future –, other than *(iii)* the many successful works on NAS leveraging on evolutionary approaches as well (cf. [23,22,46]). In particular, our evolutionary algorithm is inspired to regularised evolution proposed in [32]. However, we avoid regularisation through aging and introduce a randomised approach to explore untouched areas of the search space.

As any other evolutionary approach, our algorithm mimics biological evolution by letting a *population* of N randomly-generated NN step through a number ν of generations. More in details, the number of generations (i.e., ν) represents the maximum amount of iterations that the evolutionary algorithm should perform before returning the final solution. While transitioning between generations, NN may probabilistically *mutate*, other than being allowed to survive depending on their *fitness*. Accordingly, while the *mutation* mechanism lets the algorithm *randomly* explore different internal structures for the i^{th} cell, the *fitness measure* lets the algorithm assess how good a particular internal structure of the i^{th} cell actually is. The *Shallow2Deep* algorithm can then go on with its iteration and focus on the $(i+1)^{th}$ cell. Once reached the ν^{th} generation, the best fitting NN is used to determine the final interal structure to be chosen for the i^{th} cell.

Accordingly, in the remainder of this section, we delve into the details of how mutation and fitness actually work in the particular case of *Shallow2Deep*.

Algorithm stub. We denote by P_n the n^{th} generation of the population. Similarly, we denote by P_0 the initial population, which is randomly generated. The population size is kept fixed to throughout the local search procedure, as it is commonly done for evolutionary algorithms. In other words, for all $i \in \{1, \ldots, \nu\}$, the population P_n is such that $|P_n| = N$ and all the architectures of all networks in P_n conform to the constraints described in section 3.1.

Then, our evolutionary algorithm refines the population through 3 steps which are repeated at every generation. These steps are:

train — where all the NN in P_n are trained on (a subset of) the data set;
selection — where the NN which are not among the top-m fittest ones are removed from P_n;
incubation — where P_n is enriched with new NN – attained via mutation – aimed at replacing the ones cutted off by the selection step.

Shallow2Deep assumes the available data to be partitioned into 3 parts, namely the *training*, *validation*, and *test* sets. While the train step only leverages on the training set, the selection step evaluates the fitness measure of each network against the validation set. The test can then be used to assess the performance of the final network architecture output by *Shallow2Deep*.

Concerning the incubation step, it is aimed at helping *Shallow2Deep* both from a performance-maximisation and search-space-exploration-speed perspective. More precisely, it aims at generating new NN following two criteria:

- c networks are attained by mutating as many individuals in P_n through the application of *mutation* transformation;
- $r = N - m - c$ networks are randomly generated from the search space.

Best behaving structures mutation helps performance maximisation, enhancing the focus on those evolutionary paths that have proven to be strong in recent history of the population. Partially randomising incubation helps search space exploration as it allows the evolution to look for points in the space farther apart from previously beaten evolutionary paths.

Once all the three steps have been completed for generation n, and a new population has been attained, P_{n+1} and the evolutionary search can proceed with generation $n + 1$. The process is repeated ν times, after which the best performing local structure is considered as found.

Fitness measure. Fitness is measured on the *validation* set using the most adequate performance measure for the task at hand. Accordingly, in case the to-be-defined network targets classification tasks, accuracy or F1-score measures may be used. Conversely, in the case of regression tasks, MSE, MAE, or R^2 measures may be exploited instead.

In the particular case of image recognition tasks, classification accuracy is an adequate choice. More complex performance metrics may consider also FLOPS [39] and latency [38]. However, these are left for future works.

Mutation. The *mutation* transformation is applied to some NN – referred as the *parent* – in order to attain new architectures—called *children*. It only focuses on the internal structure of the i^{th} cell of the parent network, possibly affecting some of its blocks. In particular, we rely on two possible mutations that can be applied to the blocks of a cell (graphically depicted in fig. 4):

input mutation — a block B of the i^{th} cell is selected at random, it is detached from its previous input, and the output of either another block B' in the same cell or of the j^{th} cell as whole, with $j \in \{i-1, i-2\}$, is used as the new input of B—provided that the new connection does not introduce a loop or a cycle;

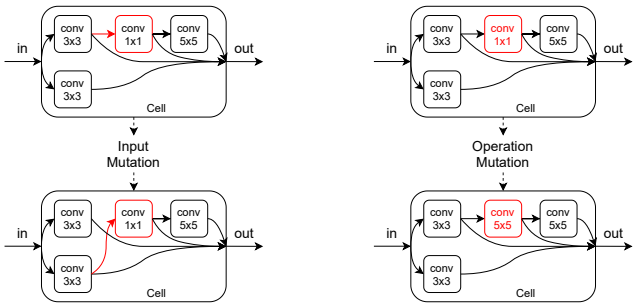

Fig. 4: Mutation operations available in randomized evolution. When input operation is applied, previous input block is linked with cell output if it has remained pendent, avoiding block removal.

operation mutation — a block B of type $o \in \mathbb{O}$ is randomly selected from the i^{th}, and its type is changed to some other $o' \in \mathbb{O}$ such that $o \neq o'$.

Greedy assemble *Shallow2Deep* requires several NN to be actually trained behind the scenes of its operation. This is true, in particular, for the evolutionary algorithm described above. In fact, while mostly focusing on one cell at a time, the algorithm must still train at least $N \cdot \nu$ networks – only differing for the content of the i^{th} cell –, C times.

To keep the computational effort feasible, a number of strategies are in place. For instance, while performing the i^{th} evolutionary search, *Shallow2Deep* leaves all cells of index j s.t. $1 \leq j < i$ unaffected, and does not re-train them anymore, as they have already been explored and trained in previous iterations. Dually, the algorithm always assumes all cells of index j s.t. $i < j \leq C$ to only contain a single block. In this way, the whole NN shallowness is preserved. In the particular case of image recognition tasks, that block may for instance consist of a 3×3 convolutional layer. Accordingly, during the i^{th} evolutionary search, only cells whose index is at least equal to i are actually trained over data, and all cells whose index is greather than i have a very minimal structure.

In other words, once the i^{th} local search is completed, the m best performing structures for the i^{th} cell are fixed, and never retrained anymore. As part of the subsequent iterations of *Shallow2Deep*, the network architecture is deepened to produce deeper and more complex NN.

4 Discussion

Global NN architectures are ideally composed by different local structures whose role depends on their position in the NN. Following this idea, unlike most common NAS frameworks, *Shallow2Deep* framework does not rely on the replication of the same cell. Rather, *Shallow2Deep* exploits a *progressive* search of the best

cells at each possible depth level, from the shallowest to the deepest ones. We here discuss the rationale behind *Shallow2Deep* progressive search.

It is well understood how the complexity of the features extracted by some NN is proportional to the depth of the layer which recognises them [47,29]. In fact, while layers that are closer to the input are appointed to extract basic features – such as edges, corners, borders, etc., in image-recognition tasks –, deeper layers aim at recognising more complex features—such as combination of shapes, combination of textures, etc. Accordingly, shallow networks are better suited to tackle simple tasks [13] where only simple features are involved. Conversely, the more complex a to-be-recognised feature is, the deeper a layer capable to recognise it must be. This happens because the recognition of a complex feature in a NN relies on the composition of more basic features extracted by shallow layers. Consequently, the lower is a feature complexity, the shallower can be the NN able to learn it. We call this phenomenon *depth-complexity proportionality* assumption.

There exists a tight link between features complexity and network depth that allows us to propose reasonable shortcuts for exhaustive architecture search methods. *Shallow2Deep* is designed on the assumption that simple features learnt by shallow networks perform reliably for deeper networks as well. Indeed, deeper NN achieve more flexible recognition capabilities than shallower ones [27,31]. Moreover, deeper NN may attain higher generalisation capabilities [34], being capable of adapting to the features that shallow NN have learnt to recognise.

Accordingly, we argue that NN built from the sequential repetition of the same local structure cannot achieve the astounding results that characterise state-of-the-art NN. Conversely, we believe it is possible to search for reliable shallow architectures and expand them in successive iterations, as done by *Shallow2Deep*. The more simple concepts are reliably learnt by shallow networks, the easier it will be to learn complex notions from their combinations. We call this phenomenon *knowledge greediness* assumption.

The progressive global assemble of *Shallow2Deep* exploits both knowledge greediness and depth-complexity proportionality assumptions to boost the overall time complexity and performance.

In particular, depth-complexity proportionality justifies the deepening of the NN architecture in successive iterations, which in turns supports the trick exploited by *Shallow2Deep* to speed up the local search phase. Indeed, population training in the evolutionary local search is the most expensive and time consuming process. Training shallower networks requires less time to complete, as the parameters to optimize are much less.

Conversely, knowledge greediness justifies *Shallow2Deep*'s strategy of iteratively expanding the depth of the NN architecture under consideration. While this certainly raises NN training time, it also lets deeper architectures rely on previously trained cells. In particular, to boost the overall search, *Shallow2Deep* fixes the parameters of shallower cells, avoiding their re-training. This idea traces back to the well-established idea of re-using pre-trained feature extractors in object detection mechanisms [33,20]. Indeed, once a network is deepened and its

deeper cell structure is determined, the predictive performance of the overall network does not degrade, even if shallow layers are kept fixed.

5 Experiments

In this section we first present *Shallow2Deep* implementation and the best NN architecture obtained with it (see Section 5.1). In Section 5.2 we then compare obtained architecture with state-of-the-art models that leverage on the same operation set $\mathbb{O}$. We also analyse if *Shallow2Deep* local structures could be reused through repetition in a NN model to obtain better performance/complexity ratio. We make publicly available our implementation of *Shallow2Deep*.[5]

5.1 *Shallow2Deep* Architecture

In order to demonstrate the validity of our approach we run *Shallow2Deep* on MNIST fashion [44]. We define $\mathbb{O}$ to be the set of available operations that can be selected for each block of a cell. Similar to [46], in *Shallow2Deep* $\mathbb{O}$ contains simple convolutions and identity (*1×1 conv*, *3×3 conv*, *5×5 conv*, *identity*). Consider now *Shallow2Deep* search space $\mathbb{S}$—i.e. the space that contains obtainable cells through local search. The search space cardinality – i.e. the number of obtainable cells – is $|\mathbb{S}| = (B+1)! \cdot |\mathbb{O}|^B$. Let now $\mathbb{N}$ be the search space for the overall NN—i.e. the set of obtainable NN architectures. Remembering that *Shallow2Deep* does not rely on cell repetition, the amount of NN architectures available during the overall architecture search is $|\mathbb{N}| = |\mathbb{S}|^C = \left((B+1)! \cdot |\mathbb{O}|^B\right)^C$. For our experiments we set $B = 3$ and $C = 4$, obtaining $|\mathbb{S}| = 1.54 \cdot 10^3$ and $|\mathbb{N}| = 5.57 \cdot 10^{12}$. The amount of possible NN architectures is huge, but it does not reflect the computational complexity. Indeed, thanks to its increasing depth approach, *Shallow2Deep* is capable of searching a space of size $|\mathbb{S}|^C$, while having complexity that is only proportional to $C \cdot |\mathbb{S}|$

For each cell we search for the best structure using the randomised evolution algorithm proposed in Section 3.2. We fix the number of generations of the evolutionary algorithm to be $\nu = 5$ for each cell and the population size to be $|P| = 50$. During *incubation* we fixed the number of surviving best models to be $m = 10$, the number of models obtained through mutations to be $c = 20$ and the number of random models added to each generation to be $r = 20$. Each model is trained for 10 epochs using learning rate $learning_rate = 0.01$.

To show the effectiveness of *Shallow2Deep* search, we study the behaviour of the NN population against the number of generations of the overall algorithm. In *Shallow2Deep*, the user can select the number of cells C that compose the NN and ν, the number of generations that the local search takes. *Shallow2Deep* iteratively searches each of the C cells for ν generations. Therefore, the overall search of the NN architecture takes $C \cdot \nu$ generations to complete. We study the average performance – i.e. classification accuracy – of the population of NN for

[5] `https://github.com/AndAgio/Shallow2Deep`

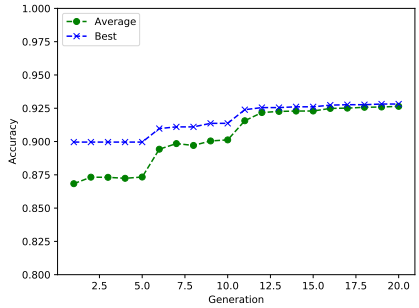

Fig. 5: Performance – i.e. classification accuracy – of NN architectures considered by *Shallow2Deep* for each generation. We consider both the average performance and the accuracy of the best model in each generation.

each of the $C \cdot \nu$ generations. We also study the accuracy of the best NN in the population for each of *Shallow2Deep* $C \cdot \nu$ generations.

Figure 5 shows the behaviour of average and best NN performance against *Shallow2Deep* generations. The classification accuracy increases with the number of generations considered, showing the success of *Shallow2Deep* search. Accuracy increments are limited since even 1^{st} generation NN reach reasonable performances. This is due to the mild complexity of the classification task over the MNIST fashion dataset. Biggest increments in the NN accuracy are found in generations where the cell index i is increased—i.e. local search shifts to the next cell. This behaviour is expected as the increasing complexity – i.e. depth – of the NN extends its reasoning capabilities. It is also interesting to notice that this behaviour is more evident for smaller cell index i, while it becomes more attenuated for values of i close to C. In our experiments, performance reaches stability for $i = C$—i.e. there exists a negligible difference between accuracy of NN with $i = C - 1$ and $i = C$. Stabilisation of accuracy can be considered a signal that the NN is reaching a complexity limit. Surpassing this limit would increase concepts complexity while not bringing any gain in performance, introducing possible overthinking issues [19]. Therefore, *Shallow2Deep* represents a tool to automatically identify the NN complexity sweetspot over a certain task.

Figure 6 shows the architecture of the NN obtained running *Shallow2Deep* on the MNIST fashion dataset.

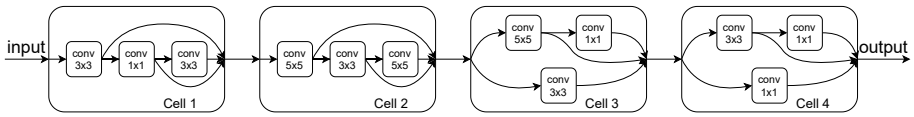

Fig. 6: NN architecture discovered by *Shallow2Deep* algorithm.

From its architecture, we point out that *Shallow2Deep* NN identifies sequential operations – i.e. blocks connected in a sequential manner inside a cell – at shallower stages of the NN—i.e. cells 1 and 2. Going deeper in the NN architecture – i.e. cells 3 and 4 –, *Shallow2Deep* building procedure identifies cells composed of parallel branches of convolutional operations. If confirmed in future investigations, this concept might give some insights on the learning process of NN. It is possible that sequential operations at shallow sections of NN help the model to learn simple concepts at the basis of their reasoning — i.e. edges, corners, simple shapes, etc. Parallel operations approach may, instead, be useful for the NN learning process when complex concepts need to be extracted—i.e. combination of shapes, combination of textures, etc. Therefore, deeper investigation of this result may be interesting.

5.2 *Shallow2Deep vs.* State-of-the-art

We now compare the performances obtained by *Shallow2Deep* NN against state-of-the-art models that apply the same basic operations – i.e. convolutions and identity – like VGG [37] and ResNet [15]. In order to make the comparison fair, we retrain the *Shallow2Deep* NN, VGG, and ResNet on the MNIST fashion dataset from scratch. Training parameters are the same for every model considered—i.e. 60 epochs and $learning_rate = 0.01$. Moreover, to study the effects of cell structure variability in NN architectures, we consider NN models built from the repetition of single cells found by *Shallow2Deep* local search. In other words, we select *Shallow2Deep* cell i – i.e. the cell discovered during i^{th} local search step – and we build the NN model composed of 4 cells having the same structure of cell i. We name these NN architectures $Shallow2Deep_i$.

Table 1 shows the performance over the test set T– i.e., the average accuracy and its standard deviation over 20 training runs –, the footprint – i.e. number of weights of the NN (expressed in millions, denoted by M) – of *Shallow2Deep* and state-of-the-art NN. *Shallow2Deep* NN with its variants reach state-of-the-art performances over the MNIST fashion classification dataset. NN obtained using *Shallow2Deep* are the most efficient if we consider the accuracy/footprint trade-off—i.e. division between reached accuracy and number of parameters. More in details, *Shallow2Deep* NN reaches accuracy comparable with VGG (only 0.4% less), while requiring a third of the parameters. Performances obtained by the ResNet NN over the dataset under examination are possibly due to overthinking issues. ResNet model complexity – i.e. model footprint – is higher than necessary for the selected task, which brings it to learn too many or too complex concepts, decreasing overall performances.

We also analyse the effects of cell structure variability in NN architectures. Base *Shallow2Deep* NN version intrinsically express high level of cell structure variability, while its variants – e.g. $Shallow2Deep_i$ – do not. It is possible to notice that *Shallow2Deep* NN outperforms 3 of its $Shallow2Deep_i$ variants out of 4 in terms of absolute performances. Moreover, *Shallow2Deep* NN outperforms all of its $Shallow2Deep_i$ variants when the accuracy/footprint trade-off is considered.

Model name	Accuracy ± std (%)	Parameters (M)
Shallow2Deep	93.26 ± 0.18	0.251
$Shallow2Deep_1$	92.87 ± 0.17	0.165
$Shallow2Deep_2$	93.31 ± 0.14	0.491
$Shallow2Deep_3$	92.73 ± 0.14	0.377
$Shallow2Deep_4$	92.28 ± 0.10	0.118
VGG	93.66 ± 0.18	0.746
ResNet	92.77 ± 0.09	1.626

Table 1: Comparison between *Shallow2Deep* and state-of-the-art models. We consider also models built through repetition of single *Shallow2Deep* cells—e.g. $Shallow2Deep_1$ is the NN built from repetition of *Shallow2Deep* cell 1 in a sequential manner.

Therefore, we can safely state that cell structure variability allows NN models to reach higher performances while being complexity-constrained.

6 Conclusion

In this work we propose *Shallow2Deep*, a novel NAS approach that limits NN complexity and promotes local variability in their architectures. *Shallow2Deep* relies on successive searches of local optima and NN expansions – i.e. depth increment – to produce well performing NN models.

We show that *Shallow2Deep* can effectively achieve NN complexity reduction, while reaching performances comparable to the state-of-the-art. Complexity reduction is tightly linked with NN opacity. Along this line, we also discuss why *Shallow2Deep* enables the application of explainability techniques to unprecedented scenarios, by providing a means to control NN structural design.

To the best of our knowledge, the proposed work represents the first approach to design an automatic tool to produce efficient NN architectures, while identifying complexity limits of NN models and helping designers to avoid overthinking issues or unnecessary opacity increments. In particular, this work represents a first approach to the analysis of structure variability influence on NN model performances, as *Shallow2Deep* promotes local variability. Along this path, our experimental analysis demonstrates how variability over local structures that compose NN is a desirable feature to obtain small and well performing models. This idea is in contrast with previously-proposed NN design approaches that neglect local structure variability, opening new possibilities for future research.

Acknowledgements

This paper has been partially supported by *(i)* the H2020 project "StairwAI" (G.A. 101017142), and *(ii)* the CHIST-ERA IV project "EXPECTATION" (G.A. CHIST-ERA-19-XAI-005).

References

1. Adadi, A., Berrada, M.: Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). IEEE Access **6**, 52138–52160 (2018). https://doi.org/10.1109/ACCESS.2018.2870052
2. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. PLOS ONE **10**(7), 1–46 (2015). https://doi.org/10.1371/journal.pone.0130140
3. Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., Herrera, F.: Explainable explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion **58**(December 2019), 82–115 (2020). https://doi.org/10.1016/j.inffus.2019.12.012
4. Bau, D., Zhou, B., Khosla, A., Oliva, A., Torralba, A.: Network dissection: Quantifying interpretability of deep visual representations. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. pp. 3319–3327. IEEE Computer Society (2017). https://doi.org/10.1109/CVPR.2017.354
5. Cai, H., Zhu, L., Han, S.: Proxylessnas: Direct neural architecture search on target task and hardware. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net (2019), `https://openreview.net/forum?id=HylVB3AqYm`
6. Calegari, R., Ciatto, G., Omicini, A.: On the integration of symbolic and sub-symbolic techniques for XAI: A survey. Intelligenza Artificiale **14**(1), 7–32 (2020). https://doi.org/10.3233/IA-190036
7. Casale, F.P., Gordon, J., Fusi, N.: Probabilistic neural architecture search. CoRR **abs/1902.05116** (2019), `http://arxiv.org/abs/1902.05116`
8. Chen, S., Bateni, S., Grandhi, S., Li, X., Liu, C., Yang, W.: DENAS: automated rule generation by knowledge extraction from neural networks. In: Devanbu, P., Cohen, M.B., Zimmermann, T. (eds.) ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020. pp. 813–825. ACM (2020). https://doi.org/10.1145/3368089.3409733, `https://doi.org/10.1145/3368089.3409733`
9. Chu, X., Zhang, B., Xu, R.: Multi-objective reinforced evolution in mobile neural architecture search. In: Bartoli, A., Fusiello, A. (eds.) Computer Vision - ECCV 2020 Workshops - Glasgow, UK, August 23-28, 2020, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 12538, pp. 99–113. Springer (2020). https://doi.org/10.1007/978-3-030-66823-5_6
10. Ciatto, G., Schumacher, M.I., Omicini, A., Calvaresi, D.: Agent-based explanations in AI: Towards an abstract framework. In: Calvaresi, D., Najjar, A., Winikoff, M., Främling, K. (eds.) Explainable, Transparent Autonomous Agents and Multi-Agent Systems, Lecture Notes in Computer Science, vol. 12175, pp. 3–20. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51924-7_1
11. Dosilovic, F.K., Brcic, M., Hlupic, N.: Explainable artificial intelligence: A survey. In: Skala, K., Koricic, M., Grbac, T.G., Cicin-Sain, M., Sruk, V., Ribaric, S., Gros, S., Vrdoljak, B., Mauher, M., Tijan, E., Pale, P., Janjic, M. (eds.) 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018, Opatija, Croatia, May 21-25, 2018. pp. 210–215. IEEE (2018). https://doi.org/10.23919/MIPRO.2018.8400040

12. Elsken, T., Metzen, J.H., Hutter, F.: Neural architecture search: A survey. Journal of Machine Learning Research **20**, 55:1–55:21 (2019), http://jmlr.org/papers/v20/18-598.html
13. Golovko, V., Egor, M., Brich, A., Sachenko, A.: A Shallow Convolutional Neural Network for Accurate Handwritten Digits Classification. In: Krasnoproshin, V.V., Ablameyko, S.V. (eds.) Pattern Recognition and Information Processing. pp. 77–85. Communications in Computer and Information Science, Springer, Cham (2017)
14. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM Computing Surveys **51**(5) (Aug 2018). https://doi.org/10.1145/3236009
15. He, K., Zhang, X., Ren, S., Sun, J.: 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. pp. 770–778. IEEE Computer Society (2016). https://doi.org/10.1109/CVPR.2016.90
16. Hecht-Nielsen, R.: Theory of the backpropagation neural network. Neural Networks **1**(Supplement-1), 445–448 (1988). https://doi.org/10.1016/0893-6080(88)90469-8
17. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR **abs/1704.04861** (2017), http://arxiv.org/abs/1704.04861
18. Janzing, D., Minorics, L., Blöbaum, P.: Feature relevance quantification in explainable AI: A causal problem. In: Chiappa, S., Calandra, R. (eds.) The 23rd International Conference on Artificial Intelligence and Statistics (AISTATS 2020). Proceedings of Machine Learning Research, vol. 108, pp. 2907–2916 (2020), http://proceedings.mlr.press/v108/janzing20a.html
19. Kaya, Y., Hong, S., Dumitras, T.: Shallow-deep networks: Understanding and mitigating network overthinking. In: Chaudhuri, K., Salakhutdinov, R. (eds.) 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, CA, USA. Proceedings of Machine Learning Research, vol. 97, pp. 3301–3310 (2019), http://proceedings.mlr.press/v97/kaya19a.html
20. Li, J., Liang, X., Shen, S., Xu, T., Feng, J., Yan, S.: Scale-aware fast R-CNN for pedestrian detection. IEEE Trans. Multim. **20**(4), 985–996 (2018). https://doi.org/10.1109/TMM.2017.2759508
21. Lipton, Z.C.: The mythos of model interpretability. Queue **16**(3), 31—57 (Jun 2018). https://doi.org/10.1145/3236386.3241340
22. Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L., Fei-Fei, L., Yuille, A.L., Huang, J., Murphy, K.: Progressive neural architecture search. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11205, pp. 19–35. Springer (2018). https://doi.org/10.1007/978-3-030-01246-5_2
23. Liu, H., Simonyan, K., Yang, Y.: DARTS: differentiable architecture search. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net (2019), https://openreview.net/forum?id=S1eYHoC5FX
24. Liu, J., Tripathi, S., Kurup, U., Shah, M.: Pruning algorithms to accelerate convolutional neural networks for edge applications: A survey. CoRR **abs/2005.04275** (2020), https://arxiv.org/abs/2005.04275
25. Luo, R., Tan, X., Wang, R., Qin, T., Chen, E., Liu, T.: Neural architecture search with GBDT. CoRR **abs/2007.04785** (2020), https://arxiv.org/abs/2007.04785

26. Miller, G.F., Todd, P.M., Hegde, S.U.: Designing neural networks using genetic algorithms. In: Schaffer, J.D. (ed.) 3rd International Conference on Genetic Algorithms. pp. 379–384. Morgan Kaufmann, Fairfax, VA, USA (Jun 1989)
27. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. pp. 4293–4302. IEEE Computer Society (2016). https://doi.org/10.1109/CVPR.2016.465
28. Nguyen, A.M., Dosovitskiy, A., Yosinski, J., Brox, T., Clune, J.: Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In: Lee, D.D., Sugiyama, M., von Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain. pp. 3387–3395 (2016), `https://proceedings.neurips.cc/paper/2016/hash/5d79099fcdf499f12b79770834c0164a-Abstract.html`
29. Nguyen, A.M., Yosinski, J., Clune, J.: Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. CoRR **abs/1602.03616** (2016), `http://arxiv.org/abs/1602.03616`
30. O'Shea, K., Nash, R.: An introduction to convolutional neural networks. CoRR **abs/1511.08458** (2015), `http://arxiv.org/abs/1511.08458`
31. Peng, S., Ji, F., Lin, Z., Cui, S., Chen, H., Zhang, Y.: MTSS: learn from multiple domain teachers and become a multi-domain dialogue expert. In: AAAI Conference on Artificial Intelligence (AAAI-20 Technical Tracks 5). vol. 34, pp. 8608–8615. AAAI Press (2020). https://doi.org/10.1609/aaai.v34i05.6384
32. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. In: AAAI Conference on Artificial Intelligence (AAAI-19, IAAI-19, EAAI-20). vol. 33, pp. 4780–4789. AAAI Press (2019). https://doi.org/10.1609/aaai.v33i01.33014780
33. Ren, S., He, K., Girshick, R.B., Zhang, X., Sun, J.: Object detection networks on convolutional feature maps. IEEE Trans. Pattern Anal. Mach. Intell. **39**(7), 1476–1481 (2017). https://doi.org/10.1109/TPAMI.2016.2601099
34. Rolnick, D., Tegmark, M.: The power of deeper networks for expressing natural functions. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net (2018), `https://openreview.net/forum?id=SyProzZAW`
35. Samek, W., Binder, A., Montavon, G., Lapuschkin, S., Müller, K.: Evaluating the visualization of what a deep neural network has learned. IEEE Trans. Neural Networks Learn. Syst. **28**(11), 2660–2673 (2017). https://doi.org/10.1109/TNNLS.2016.2599820
36. Setiono, R., Leow, W.K., Zurada, J.M.: Extraction of rules from artificial neural networks for nonlinear regression. IEEE Transactions on Neural Networks **13**(3), 564–577 (2002). https://doi.org/10.1109/TNN.2002.1000125
37. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), `http://arxiv.org/abs/1409.1556`
38. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: Mnasnet: Platform-aware neural architecture search for mobile. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019. pp. 2820–2828. Computer Vision Foundation / IEEE (2019). https://doi.org/10.1109/CVPR.2019.00293

39. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks. CoRR **abs/1905.11946** (2019), http://arxiv.org/abs/1905.11946
40. Thrun, S.: Extracting rules from artificial neural networks with distributed representations. In: 7th International Conference on Neural Information Processing Systems (NIPS'94). pp. 505–512. MIT Press (1994)
41. Tjoa, E., Guan, C.: A survey on explainable artificial intelligence (XAI): towards medical XAI. CoRR **abs/1907.07374** (2019), http://arxiv.org/abs/1907.07374
42. Wistuba, M., Rawat, A., Pedapati, T.: A survey on neural architecture search. CoRR **abs/1905.01392** (2019), http://arxiv.org/abs/1905.01392
43. Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., Keutzer, K.: Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019. pp. 10734–10742. Computer Vision Foundation / IEEE (2019). https://doi.org/10.1109/CVPR.2019.01099
44. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. CoRR **abs/1708.07747** (2017), http://arxiv.org/abs/1708.07747
45. Yang, Z., Wang, Y., Chen, X., Shi, B., Xu, C., Xu, C., Tian, Q., Xu, C.: CARS: continuous evolution for efficient neural architecture search. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020. pp. 1826–1835. IEEE (2020). https://doi.org/10.1109/CVPR42600.2020.00190
46. Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., Hutter, F.: Nas-bench-101: Towards reproducible neural architecture search. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA. Proceedings of Machine Learning Research, vol. 97, pp. 7105–7114. PMLR (2019), http://proceedings.mlr.press/v97/ying19a.html
47. Yosinski, J., Clune, J., Nguyen, A.M., Fuchs, T.J., Lipson, H.: Understanding neural networks through deep visualization. CoRR **abs/1506.06579** (2015), http://arxiv.org/abs/1506.06579
48. Zhang, Q., Wu, Y.N., Zhu, S.: Interpretable convolutional neural networks. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018. pp. 8827–8836. IEEE Computer Society (2018). https://doi.org/10.1109/CVPR.2018.00920
49. Zhang, Q., Yang, Y., Ma, H., Wu, Y.N.: Interpreting cnns via decision trees. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019. pp. 6261–6270. Computer Vision Foundation / IEEE (2019). https://doi.org/10.1109/CVPR.2019.00642
50. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. In: 5th International Conference on Learning Representations (ICLR 2017). Toulon, France (April 24-26 2017), https://openreview.net/forum?id=r1Ue8Hcxg