Automated Machine Learning: Prospects and Challenges

Lorenzo Vaccaro, Giuseppe Sansonetti(✉), and Alessandro Micarelli

Department of Engineering, Roma Tre University,
Via della Vasca Navale 79, 00146 Rome, Italy
lor.vaccaro1@stud.uniroma3.it, {gsansone,micarel}@dia.uniroma3.it

Abstract. The State of the Art of the young field of Automated Machine Learning (AutoML) is held by the connectionist approach. Several techniques of such an inspiration have recently shown promising results in automatically designing neural network architectures. However, apart from back-propagation, only a few applications of other learning techniques are used for these purposes. The back-propagation process takes advantage of specific optimization techniques that are best suited to specific application domains (e.g., Computer Vision and Natural Language Processing). Hence, the need for a more general learning approach, namely, a basic algorithm able to make inference in different contexts with distinct properties. In this paper, we deal with the problem from a scientific and epistemological point of view. We believe that this is needed to fully understand the mechanisms and dynamics underlying human learning. To this aim, we define some elementary inference operations and show how modern architectures can be built by a combination of those elementary methods. We analyze each method in different settings and find the best-suited application context for each learning algorithm. Furthermore, we discuss experimental findings and compare them with human learning. The discrepancy is particularly evident between supervised and unsupervised learning. Then, we determine which elementary learning rules are best suited for unsupervised systems, and, finally, we propose some improvements in reinforcement learning architectures.

Keywords: Automated Machine Learning · Meta Learning · Neural Architecture Search · Reinforcement learning

1 Introduction and Background

Artificial Intelligence (AI) is increasingly part of our lives. It organizes the services of our cities [7], suggests which points [28] are likely to be of interest to us (e.g., artistic and cultural resources [29] or restaurants [2]) and how to reach them [11]. It recommends us which news articles [4] or research papers [16] to read, which movies to watch [1], which products to buy [3], which music artists and songs to listen to [26], and even which people to attend [10]. To do this,

© Springer Nature Switzerland AG 2020
O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12252, pp. 119–134, 2020.
https://doi.org/10.1007/978-3-030-58811-3_9

however, more and more efficient *algorithms* are required, to obtain which we need an ever-deeper understanding of the theory behind them.

David Hilbert [18], Alan Turing [32], and Alonzo Church [6] were the first to formalize logic and automatic reasoning through algorithms. Since then, human and machine learning have been studied, analyzed, and classified. The first criterion with which we will analyze AI algorithms is a classification of the reasoning process into two categories. Such a criterion is reflected in Psychology and AI. In particular, two types of reasoning were found in the functions associated with the various areas and components of the brain: *top-down* and *bottom-up*. They were associated with the perception of sensory inputs and the imagination of these, respectively. In [17], the authors hypothesize that the activity of the central nervous system reflects the process of bringing together internally generated predictions and external sensory perceptions. According to this theory, our brain learns by generating predictions and comparing them with the perceived reality. In AI, however, bottom-up reasoning refers to the process of inferring a value, a function, or an algorithm through a learning method. A series of *input-output* example pairs is provided and the program searches for the algorithms that relate them. Differently, with the top-down approach, we designate the algorithms coded with a priori knowledge of the problem. The reasoning is implemented in a well-defined algorithm that transforms inputs into outputs. Automated Machine Learning (AutoML) is the process that autonomously reshapes top-down knowledge into bottom-up reasoning.

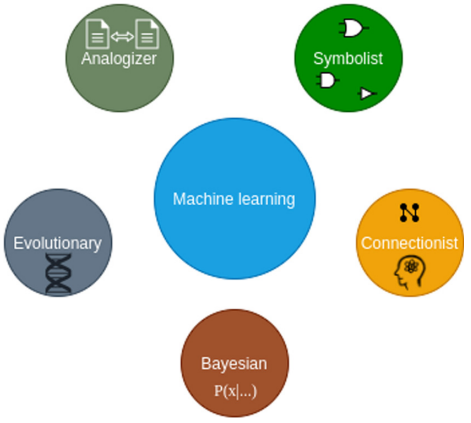


Fig. 1. The five paradigms of Machine Learning.

To classify the Machine Learning (ML) methods, we follow the following definition that incorporates the one proposed in [8] (see Fig. 1). It starts by defining the five lines of thinking that form the basis for ML methods. Pedro Domingos defines the *master algorithm* as the algorithm behind learning and identifies this component for the five paradigms. For each of them, it defines how information is represented and inferred. In the *symbolist* paradigm, the rigorous syntax

and the specific representation adopted make it difficult to model the learning method. In this paradigm, it is easy to understand and model bottom-up reasoning and apply it to learning itself. However, the efficiency of these systems is highly dependent on the specific formulation. The *connectionist* paradigm has recently gained the interest of many researchers. The main advantage of these architectures is efficiency. In the training phase of neural networks, the parameters are inferred through bottom-up reasoning by induction. The trained network can quickly classify the test samples through deductive top-down reasoning. In these architectures, the structure choice is fundamental and directly shapes the limits of the solution complexity. The Neural Architecture Search (NAS) [9] science shows an example of learning application to the search for neural architectures through the connectionist paradigm. This example shows that connectionist learning can be applied to itself. However, the most successful NAS models jointly apply different learning paradigms. One of the major limitations of connectionist approaches is the modeling of discrete values and ordinal variables. The *Bayesian* paradigm introduces uncertainty. This powerful tool allows for a more realistic representation of learning. It also enables the direct modeling of understanding. It independently carries out the distinction between known and uncertain information. Among the hybrid learning methods applied to the research on neural architectures, the Bayesian paradigm has proven to be very efficient. The *evolutionary* paradigm is likely to be the most complete for the potential proposed solutions. This method is the most permissive for defining the solution space. For this reason, it generally takes more time and is subject to a higher risk of getting stuck in a local minimum. Compared to the one proposed in [8], our definition of ML paradigms differs as follows. Domingos attributes the *analogizer* paradigm only to kernel machines. Differently, we consider it present in every learning method that applies reasoning by analogy. Any ML algorithm that applies a direct or indirect similarity measure between two elements can be considered part of the analogizer paradigm as it introduces a similarity criterion.

The rest of this paper is structured as follows. In Sect. 2, the three experimental phases that constituted our research path are described in detail. In particular, the first phase focused on the implementation of an inference model based on the symbolist paradigm. In the second phase, the search algorithms for neural architectures were analyzed. The search for neural architectures was then compared with other known ML problems. In the third experimental phase, an architecture including some hybrid learning methods was proposed. During this phase, three ML models were deeply analyzed, which introduce inference through reasoning by analogy in connectionist systems. In Sect. 3, we draw our conclusions and outline some possible developments in our research activities.

2 The Experimental Path

2.1 First Experimental Phase

The first experimental phase focused on defining a symbolist model. The proposed architecture was implemented starting from an article[1] in which evolutionary learning is applied to the symbolist paradigm. This article describes an experiment that leverages an *evolutionary algorithm* to generate a Turing Machine able of correctly transforming the empty input into the requested output. To simulate the behavior of a Turing Machine without however having to generate code in a verbose programming language, the article proposes to use the BrainF*ck (BF) language designed in 1993 by Urban Müller. A useful feature of BF is the very small set of instructions sufficient to make the language Turing complete. BF is an esoteric language, it is difficult to program, however, both the instructions and the code have a common format: the ASCII coding. The input, the output, and the code are strings and the instructions consist of a single character. To avoid having to also generate the keyboard output, the "," character was removed from the instructions in the experiment of the aforementioned article. Also for this experiment, the output of the program is considered what it prints, not what remains on the tape at the end of the program. To demonstrate the potential of this system, the article shows some experiments that generate words or sentences in English. This forced approach allows for limited efficacy, several generations in the order of magnitude of one million are needed to generate a program capable of writing a sentence. In general, applying random generation to code increases the risk of running into syntax errors. To improve the ability of the system to generate the correct code, we considered several alternatives. To check the code it is sufficient to use a type-2 grammar, to generate the code we need a pushdown automaton (PDA).

$$S \rightarrow SS|+|-|<|>|[S]|\varepsilon$$

The proposed model generates solutions of increasing complexity, sequentially, until a valid solution is reached. Instead of randomly drawing a new instruction at each step, as in the aforementioned article, the algorithm explores breadth-first the tree of possible codes. In this tree, each node represents a different code. Each arc represents a choice (using the type-2 grammar of the original BF language is an unbalanced 7-ary tree). The height of a node in the tree is a direct measure of the code complexity. Exploring the tree of possible programs is an algorithm. In this model, there is no distinction between code and data. The relationship between input and output is treated in the same way as the relationship between the previously generated code and the current code. The architecture is organized in layers (see Fig. 2). All information inferred into a layer is coded and learned from the upper level. From this starting point, we experimented with different ways of evolving the model. In particular, we paid attention to trying

[1] www.primaryobjects.com/2013/01/27/using-artificial-intelligence-to-write-self-modifying-improving-programs/ (Accessed: June 23, 2020).

to evolve the architectures *vertically*, to ensure that the inference process took place upstream of the pile of architectures. The reason is simple: if we consider an architecture capable of generating the algorithm of the sum, for example, we can expect that this architecture is also capable of generating the code for subtraction. A higher layer architecture, if properly trained, can generalize learning outside the problems already addressed. Alternatively, it is also possible to make inference in the model by backtracking the code. The hypothesis of such a system has already been formulated by Jürgen Schmidhuber in his paper entitled "Optimal Ordered Problem Solver (OOPS)" [30]. In this experimental phase, we focused on the search for learning algorithms capable of scaling in complexity. The architecture expands dynamically, with each step the original problem is converted into the search for a learning algorithm specific to that problem. The dimensionality explosion of the research space introduces a computation overhead that makes the method inefficient.

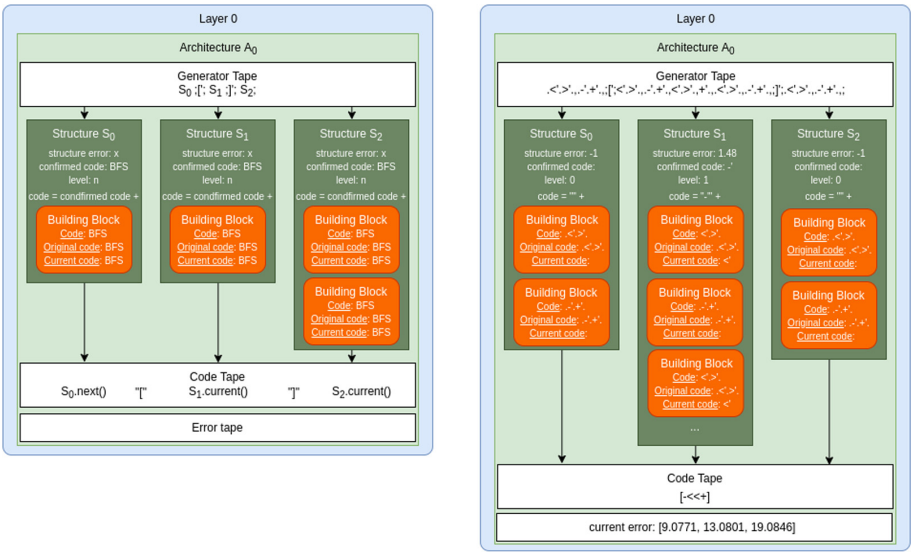


Fig. 2. High-level scheme of a hierarchically organized generator architecture.

2.2 Second Experimental Phase

In the second experimental phase, we collected and tested many NAS algorithms. This discipline proposes a connectionist approach to the problem of learning to learn. The different architectures are distinguished by the upstream learning process. With this method, the parameters of the underlying connectionist model are inferred. Early attempts model the system as a reinforcement learning (RL) problem [33]. The problem definition was then mathematically formalized.

The search space is composed of all the combinations of possible parameters of the connectionist network to be modeled. Starting from this definition, in the Bayesian paradigm Autokeras [22] and NASBOT [23] are proposed. Evolutionaries contribute with Hierarchical Evo [24] and AmoebaNet [27]. We have not found any symbolist NAS applications. For the analogizer paradigm, on the other hand, we have found a strong relationship with the most successful algorithms in this discipline. The first problem we faced in the application of connectionist techniques to NAS is raised by the discrete nature of the research space. This problem, common also to unsupervised and reinforcement learning, is solved by techniques attributable to reasoning by analogy. Among the most common and exhaustively studied applications, which require an elementary inference mechanism similar to that necessary for the search for neural architectures, we find automatic text translation, game solving, and many other complex tasks. Therefore, in the subsequent experimental phase, we analyzed RL architectures and some models that jointly exploit the connectionist, Bayesian, and analogizer paradigms.

2.3 Third Experimental Phase

The initial goal of the third experimental phase was to overcome the limitations of the connectionist models in the application context of complex tasks. In particular, we found two models of interest in this respect: Differentiable Neural Computer (DNC) [13] and Neural Arithmetic Logic Units (NALU) [31]. Linear activation functions are known to produce linear output. It is also known that to overcome this limitation it is possible to adopt non-linear functions with properties that make derivation simple. However, it is less known that these functions introduce another limitation. Neural networks tend not to generalize for values outside the range of training examples. If an artificial neural network is trained with data contained within a range to approximate a certain algorithm, such as the sum of its inputs, it will not be able to generalize this sum outside that range. Figure 3 shows the validation error for the addition algorithm with various activation functions. NALU was created to remedy this limitation. The concept behind this architecture is to model a set of arithmetic operations on the inputs and to assign each one a *learning gate*. In our first experiment on this model, we introduced a *bias* (see Fig. 4). Through this modification, it is possible to carry out several further operations. For example, the system without this foresight would not be able to double the input or add it to a constant. Starting from this model, we wondered why in the article they had not generalized the concept of exponential and logarithmic space. In practice, moving from a representation of the inputs in one space to the next logarithmic space, the effect of the addition operation is first elevated to the product, then to power, to power towers, and so on. If the gate could model an arbitrary integer n, apply the operations to the n-th logarithmic space, and raise to power the results n times, we could model any arithmetic operation and approximate any function. Looking for an architecture that models n as a discrete value, we come across the problems of reinforcement learning. However, there is a continuous mathematical function that models n

with continuous values: *tetration*. Tetrations are repeated or iterative exponentiation functions. The definition we adopted in our solution is iteratively defined as follows:

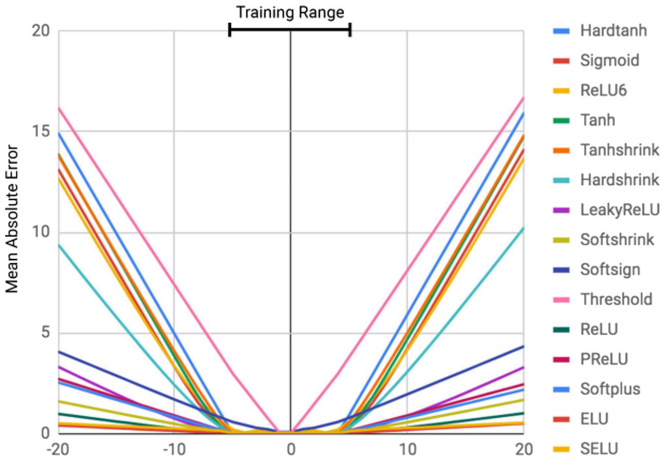


Fig. 3. The generalization problem outside the range of the training values [31].

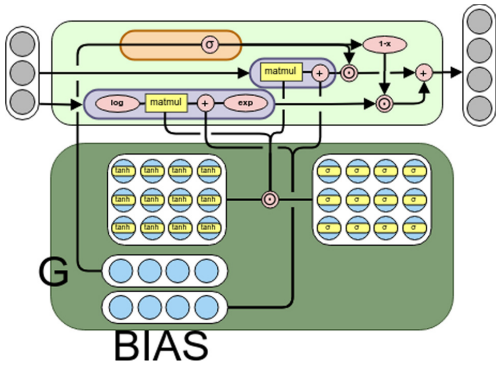


Fig. 4. First model: NALUB.

$$ {}^{x}a \approx \begin{cases} \log_a \left({}^{x+1}a\right) & x \leq -1 \\ 1 + \frac{2\ln(a)}{1+\ln(a)}x - \frac{1-\ln(a)}{1+\ln(a)}x^2 & -1 < x \leq 0 \\ a^{\left({}^{x-1}a\right)} & x > 0 \end{cases} $$

Experiments have shown that using this function always causes the algorithm to diverge. However, we tried to influence the choice of the parameter

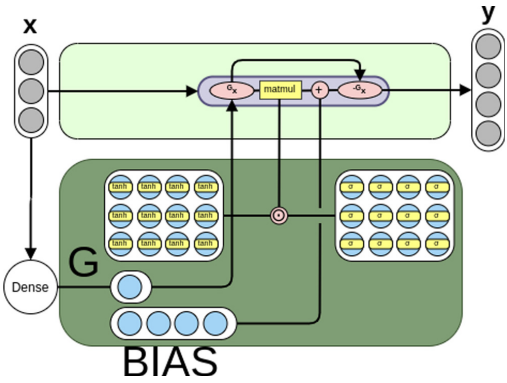


Fig. 5. Second and third model: NALUBTC.

n of the network so that the results remain correct. A valid alternative found was to approximate this parameter through a simple artificial neural network. The network also proved to be useful in alleviating the task of the NALU learning gates, leading it to converge in acceptable times. The controlled model that makes use of the tetration function and includes the bias is shown in Fig. 5. As a last experiment on the models of this architecture, we tried to define a recurrent neural network (RNN) that adopted NALU as a cell. In the scheme of the long short-term memory (LSTM) [21], we have replaced the activation functions with NALUBTC cells. We did not obtain competitive results, so we tried to integrate this model into a DNC. One of the first experiments was the implementation of a DNC grafted into another DNC. The internal network (DNC1) is the controller of the external DNC (DNC2). Let us observe the behavior of this architecture: the input is provided to the DNC1 RNN controller, this returns a set of read and write keys, the DNC1 accesses and writes into memory using the key as the instructions. The output of DNC1 is made up of the concatenation of the read cells and the reading keys. Being the output of the controller, this is interpreted by DNC2 as a set of read and write keys. In turn, DNC2 accesses the memory and returns the concatenation of the readings and output of DNC1. In this way, an indirect addressing system can be implemented, however not necessarily biunivocal. In particular, an output of DNC1 (the reading key for DNC2) points to a set of cells, a portion of memory. Therefore, while DNC2 infers several specific problem-solving algorithms, the DNC1 controller will tend to implement a problem classification algorithm. This model works particularly well for tasks where complex structures need to be modeled. In the research literature, it has been shown that DNCs can be used for information storage, graph research, Natural Language Processing (NLP), and text understanding and reasoning (bAbI Task[2]). Through the grafting of DNCs, we managed to improve the accuracy in the bAbI task, however, this increase was at the expense of effi-

[2] research.fb.com/downloads/babi/ (Accessed: June 23, 2020).

ciency. The grafted DNC model introduces a computation overhead that slows the convergence of the architecture. As for reinforcement learning, the slowdown is even more evident, especially if we compare the computational times with the classic architectures. We believe that the reason for this inefficiency lies in the fact that the RL models do not need a particular computational complexity, often the architectures with simple policies work better. In other terms, the best RL algorithms return the simplest possible solutions through complex inference systems. Furthermore, it is not clear to us which role the analogizer paradigm plays in the inference of politics. If the reasoning by analogy can influence the choice of policy, we would have expected a positive result for the integration of DNC into a classic architecture. To eliminate any doubt, we studied the ML methods that adopt reasoning by analogy, trying to figure out why it cannot be applied in certain circumstances. The initial experimentation of this research phase is present in the GitHub repository of the research work[3]. In the third experimental phase, we identified three ML mechanisms, commonly used in the aforementioned applications of automatic text translation and learning by reinforcement, which are at least in part the result of reasoning by analogy. The attention mechanism suggests an "algorithmic path" to the network on which it is applied. It hides unnecessary information and highlights useful information. By influencing the inference in the underlying network, this method could also be considered a *routing* algorithm. Routing in neural networks generally occurs on computation, in particular, it can help decide whether or not to move through a sub-network of the network. Under the forced assumption that attention is a routing algorithm, routing occurs for inputs, not on edges.

Much research has been done on the application of routing in neural networks. In particular, a publication caught our attention because of the simplicity, efficiency, and accuracy of the proposed method. Geoffrey Hinton (known also for his collaboration with Yoshua Bengio and Yann LeCun in founding *deep learning*) and his colleagues published a paper entitled "Transforming Auto-encoders" [19] that has been very successful in Computer Vision (CV). The architecture proposed by Hinton *et al.* makes use of *capsules*. They suggest that artificial neural networks must use local capsules that perform complex computations on their inputs and encapsulate the result of these computations in compact and highly informative vectors. Each capsule learns to recognize an implicitly defined graphic entity on a limited visual domain. The proposed architecture is called *CapsNet* because of the capsules that define the hierarchical structure and the relationships between the objects in the scene. The process of breaking down an image into graphic sub-components is called *inverse 3D rendering*. From the capsules of the final layer, the vectors can be used to reconstruct the images. CapsNet introduces an auxiliary *decoder* responsible for the rendering of the capsules. This foresight, borrowed from the Generative Adversarial Network (GAN) [12] architecture, allows us to keep high the coupling between the capsules and the relative representations of the entities in the image. This type of routing is called *routing by agreement* underlining the inclusion/composition

[3] github.com/lorenzoviva/tesi/tree/master/recurrent/ (Accessed: June 23, 2020).

relationship between the capsules of consecutive layers. The article points out how much this method should be more effective than the more primitive form of *max-pooling* routing implemented by modern CV algorithms. Furthermore, the article shows the first successful application to the classification of "highly overlapping objects" (images with different shapes and overlapping figures). Through the routing by agreement, it is possible to model the inclusion/composition relationships among the searched objects. It is possible to recognize and model hierarchies in the structure of examples. Therefore, it becomes clear that this type of architecture can be used to model any hierarchical structure, including the structure of software, algorithms, or choices in an unsupervised system. The concept of *dynamic routing* is not new in connectionist models. The paper entitled "Deciding How to Decide: Dynamic Routing in Artificial Neural Networks" [25] shows a dynamic routing model in neural networks. Routing in networks, like the attention mechanism [15], allows for the definition of functional units specialized in a specific task, by dividing the problem into several sub-problems with analogizer techniques and solving the sub-problems through the connectionist learning method. We were interested in the possibility of using the capsules to model the actions of an RL system. In this architecture, the solution is provided with a set of additional information, a sort of explanation of the solution. The output vector maps the distinctive properties of possible solutions. These properties are recognized in all the subcomponents of the chosen solution (e.g., in Computer Vision they can be orientation, pose, scale, or others). Capsule networks have had some success recently, inspiring researchers to develop further models [5,14,20]. In an RL system, solutions are actions, by modeling them through a vector in a network of capsules it provides us with distinctive properties that represent explanations of the solution. If the final capsules represent the action, the daughter capsules, to be recognized as belonging to the solution, must represent the motivation behind a choice. Following this logic, we tried different approaches, each of which mainly differs in the *decoder* component. The interpretation of the solution varies according to how this component is defined. In the first experimental phase, we introduced the routing mechanism in the DQN algorithm for the game environment *CartPole-v1* (from OpenAI Gym[4]). Figure 6 shows the high-level architecture of this experiment. Specifically, we used a capsule network as a policy, then we assigned the decoder the task of reconstructing the *transaction* that follows a choice. The system receives as input an image section of the current frame in the game containing the pole to be balanced, this goes through a convolutional network, a network of capsules, and the capsule related to the chosen action is returned. Starting from the capsule, the decoder reconstructs the difference between the image of the current frame and the next one. Figure 7 describes some steps of the experiment. The results show that this architecture cannot converge. As it was conceived, this architecture models only one step of the environment, it is difficult to encapsulate information that is distributed over multiple transactions. Therefore, it models a very limited prediction of action consequences. We tried several alternatives in modeling transitions (see Fig. 8).

[4] gym.openai.com/ (Accessed: June 23, 2020).

Later, we changed the DQN model replacing it with the Advantage Actor-Critic (A2C) architecture (see Fig. 9). The main advantage introduced by this architecture is to distinctly model the potential reward value associated with the state and the potential gain associated with the actions. Again, we tried to associate a capsule of the final layer with each action. We also tried several other capsule models but, among all, one method stood out for its effectiveness. The potential of the state is modeled in a single capsule of the final layer, the vector instead represents the action potentials. In a subsequent model, we introduced a further dense layer that is applied to the capsule and returns the action (see Fig. 10). The capsule vector is also used to reconstruct the original image through a decoder. This caution pushes the network to encode a system state in the vector and not to lose useful information content. In the GitHub repository of our research work, a folder (RL_routing[5]) stores the code of these models. Results in a graphical format of many experiments are also saved on Google Drive[6]. For some experiments, we also saved a dynamic image, in GIF format, which displays the capsule values over the course of an episode. The model proposed in the third experimental phase obtained competitive results with the State of the Art in learning by reinforcement. However, the motivation that encouraged us to model the choices of an agent through a network of capsules has not been reflected in practice. The most successful formulations are those that do not model the inclusion/composition relationship of the system choices through the capsules. From an intuitive point of view, it is difficult to explain the effectiveness of this method. The capsule returned by the system is a vector of magnitude equal to the state potential and the orientation defined in the action space.

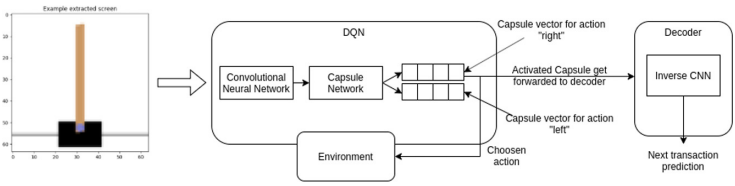
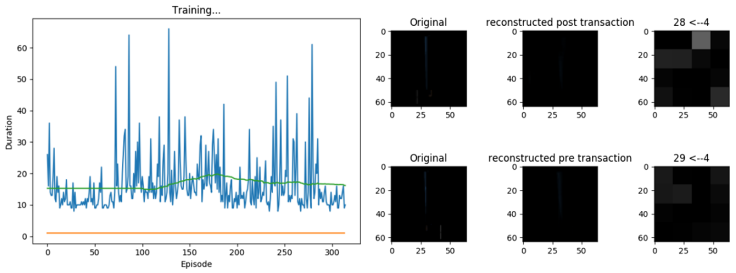


Fig. 6. A diagram showing the high-level functioning of the routing-based reinforcement learning architecture implemented on the DQN model.

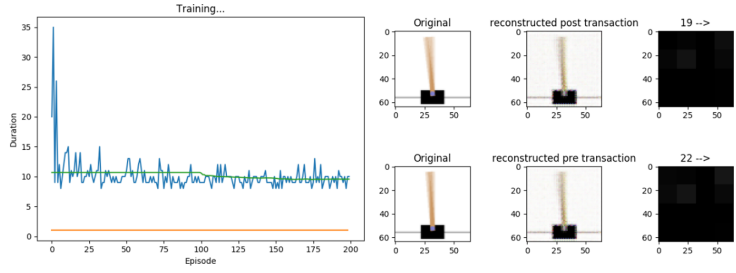
[5] http://www.github.com/lorenzoviva/tesi/tree/master/RL_routing/ (Accessed: June 23, 2020).

[6] drive.google.com/drive/folders/1n74hoJ1K0hg0SQc18y7PCH1h9w6dqFJP?usp=sharing (Accessed: June 23, 2020).



(a) Temporal trend. (b) Frames.

Fig. 7. (a) A graph showing the duration of the experiments in steps for the *CartPole-v1* environment during the training for the first experiment on routing applied to reinforcement learning. (b) On the top left the current transaction, in the center the prediction of the transition and on the right the four capsules of the first layer each consisting of four values. The first row displays the current transaction, the second row represents the previous transaction.



(a) Temporal trend. (b) Frames.

Fig. 8. A different representation for transactions.

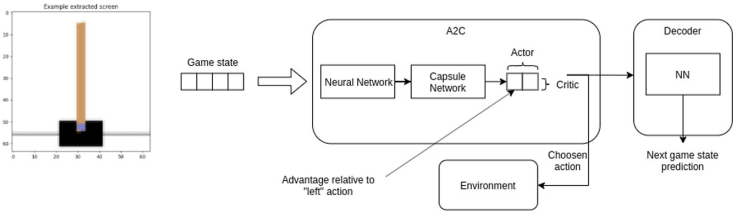


Fig. 9. A diagram showing the high-level functioning of the first reinforcement learning architecture with routing implemented on the A2C algorithm.

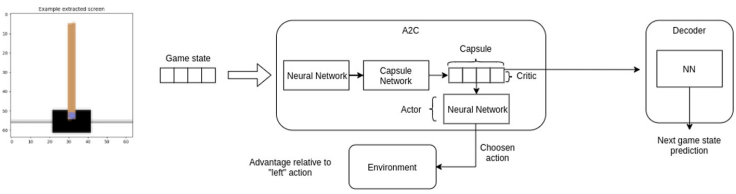


Fig. 10. A diagram showing the high-level functioning of the second reinforcement learning architecture with routing implemented on the A2C algorithm.

3 Conclusions and Future Works

The ultimate goal of our research work was to define new methods and techniques to improve the state-of-the-art models in the emerging fields of Automated Machine Learning (AutoML) and Meta Learning. Initially, we adopted a naïve approach, then we formalized the conditions for achieving the objective and, finally, we proceeded under the point of view of the *master algorithm* [8]. In the first experimental phase, we analyzed several models of the symbolist paradigm. We theorized a critical point in the development of a General Artificial Intelligence and a definitive master algorithm. We formulated criteria that allowed us to evaluate whether this goal could be achieved by certain inference algorithms. In the implementation of the symbolist ML models, we tried to structure the inference process as defined by our theory. In the development of this architecture, we introduced additional master algorithms, namely, different approaches to the inference that highlight complementary learning properties. The structural complexity and the solution specificity did not allow us to compare this system with other experiments. The symbolist models depend on the specific representation chosen. The domain in which the solutions of this method are defined is unique to this paradigm. In order to compare the experimental results, we left the symbolist paradigm for the connectionist models. Therefore, the subsequent experimentation phase focused on Meta Learning in artificial neural networks. More specifically, we experimented with different models in the Machine Learning branch called Neural Architecture Search (NAS). We compared several NAS algorithms and analyzed how they work. Finally, we tried to apply new methods and techniques to the architecture that seemed more promising to us (i.e., AutoKeras). This experimental phase led us to conclude that the connectionist paradigm alone is not sufficient to solve the NAS problem. It also allowed us to recognize that the reinforcement learning (RL) method has already addressed the problem of modeling complex dynamic structures. In the third experimental phase, we looked for learning characteristics and properties that could scale in complexity to model complex systems such as the Neural Architecture Search. In particular, we found the RL approach interesting from this point of view. In this phase, we integrated, combined, and tested RL models based on each of the five paradigms of Machine Learning. We managed to obtain competitive results with the State of the Art by applying the analogizer and the

connectionist paradigms together. A possible key for reading our results is that the combination of such paradigms makes reasoning complex enough to model the learning of the system itself.

As a first future development, we wish to focus on a framework for General Artificial Intelligence which would allow us to express all the other learning algorithms in a unique, unambiguous, formal language capable of modeling uncertainty. Pedro Domingos' Markov Logic Networks could represent a good starting point for the development of this framework. We also plan to work on the alternative use of learning in the evolutionary paradigm, not for the definition of structures, but the specification of learning algorithms. This paradigm is successful in the definition of structures since it can model the presence of discrete variables better than the connectionist paradigm. Therefore, our idea is to extend the originality of the solutions proposed by the evolutionary paradigm to the definition of algorithms. Further future development could concern reinforcement learning techniques. More specifically, after successfully applying the routing to reinforcement learning, we are currently exploring a Machine Learning branch known as Auxiliary Learning. In this application scenario, we believe that the system we designed (A2C with routing) may provide noteworthy results since the capsule routing allows us to model the hierarchical relationships present in these systems.

References

1. Biancalana, C., Gasparetti, F., Micarelli, A., Miola, A., Sansonetti, G.: Context-aware movie recommendation based on signal processing and machine learning. In: Proceedings of the 2nd Challenge on Context-Aware Movie Recommendation, CAMRa 2011, pp. 5–10. ACM, New York (2011)
2. Biancalana, C., Gasparetti, F., Micarelli, A., Sansonetti, G.: An approach to social recommendation for context-aware mobile services. ACM Trans. Intell. Syst. Technol. **4**(1), 10:1–10:31 (2013)
3. Bologna, C., De Rosa, A.C., De Vivo, A., Gaeta, M., Sansonetti, G., Viserta, V.: Personality-based recommendation in e-commerce. In: CEUR Workshop Proceedings, vol. 997. CEUR-WS.org, Aachen (2013)
4. Caldarelli, S., Gurini, D.F., Micarelli, A., Sansonetti, G.: A signal-based approach to news recommendation. In: CEUR Workshop Proceedings, vol. 1618. CEUR-WS.org, Aachen (2016)
5. Choi, J., Seo, H., Im, S., Kang, M.: Attention routing between capsules. In: Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW), pp. 1981–1989 (2019)
6. Church, A.: An unsolvable problem of elementary number theory. Am. J. Math. **58**(2), 345–363 (1936)
7. D'Aniello, G., Gaeta, M., Orciuoli, F., Sansonetti, G., Sorgente, F.: Knowledge-based smart city service system. Electronics (Switzerland) **9**(6), 1–22 (2020)
8. Domingos, P.: The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World. Basic Books, New York (2015)
9. Elsken, T., Metzen, J., Hutter, F.: Neural architecture search: a survey. J. Mach. Learn. Res. **20**, 1–21 (2019)

10. Feltoni Gurini, D., Gasparetti, F., Micarelli, A., Sansonetti, G.: Temporal people-to-people recommendation on social networks with sentiment-based matrix factorization. Future Gener. Comput. Syst. **78**, 430–439 (2018)
11. Fogli, A., Sansonetti, G.: Exploiting semantics for context-aware itinerary recommendation. Pers. Ubiquit. Comput. **23**(2), 215–231 (2019). https://doi.org/10.1007/s00779-018-01189-7
12. Goodfellow, I., et al..: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680 (2014)
13. Graves, A., et al.: Hybrid computing using a neural network with dynamic external memory. Nature **538**(7626), 471–476 (2016)
14. Hahn, T., Pyeon, M., Kim, G.: Self-routing capsule networks. In: Advances in Neural Information Processing Systems, pp. 7656–7665 (2019)
15. Hassan, H.A.M., Sansonetti, G., Gasparetti, F., Micarelli, A.: Semantic-based tag recommendation in scientific bookmarking systems. In: Proceedings of ACM RecSys 2018, pp. 465–469. ACM, New York (2018)
16. Hassan, H.A.M., Sansonetti, G., Gasparetti, F., Micarelli, A., Beel, J.: BERT, ELMo, USE and infersent sentence encoders: the panacea for research-paper recommendation? In: Tkalcic, M., Pera, S. (eds.) Proceedings of ACM RecSys 2019 Late-Breaking Results, vol. 2431, pp. 6–10 (2019). CEUR-WS.org
17. Heekeren, H.R., Marrett, S., Ungerleider, L.G.: The neural systems that mediate human perceptual decision making. Nat. Rev. Neurosci. **9**(6), 467–479 (2008)
18. Hilbert, D.: Die grundlagen der mathematik. In: Die Grundlagen der Mathematik, pp. 1–21. Springer, Wiesbaden (1928). https://doi.org/10.1007/978-3-663-16102-8
19. Hinton, G.E., Krizhevsky, A., Wang, S.D.: Transforming auto-encoders. In: Honkela, T., Duch, W., Girolami, M., Kaski, S. (eds.) ICANN 2011. LNCS, vol. 6791, pp. 44–51. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21735-7_6
20. Hinton, G.E., Sabour, S., Frosst, N.: Matrix capsules with EM routing. In: International Conference on Learning Representations (2018)
21. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
22. Jin, H., Song, Q., Hu, X.: Auto-Keras: an efficient neural architecture search system. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1946–1956 (2019)
23. Kandasamy, K., Neiswanger, W., Schneider, J., Poczos, B., Xing, E.P.: Neural architecture search with Bayesian optimisation and optimal transport. In: Advances in Neural Information Processing System, vol. 31, pp. 2016–2025. Curran Associates, Inc. (2018)
24. Liu, H., Simonyan, K., Vinyals, O., Fernando, C., Kavukcuoglu, K.: Hierarchical representations for efficient architecture search. In: Proceedings of the 6th International Conference on Learning Representations (ICLR), Vancouver, BC, Canada (2018)
25. McGill, M., Perona, P.: Deciding how to decide: dynamic routing in artificial neural networks. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 2363–2372 (2017). JMLR.org
26. Onori, M., Micarelli, A., Sansonetti, G.: A comparative analysis of personality-based music recommender systems. In: CEUR Workshop Proceedings, vol. 1680, pp. 55–59. CEUR-WS.org, Aachen (2016)

27. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. In: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI), 27 January–1 February 2019, Honolulu, Hawaii, USA, pp. 4780–4789 (2019)
28. Sansonetti, G.: Point of interest recommendation based on social and linked open data. Pers. Ubiquit. Comput. **23**(2), 199–214 (2019). https://doi.org/10.1007/s00779-019-01218-z
29. Sansonetti, G., Gasparetti, F., Micarelli, A., Cena, F., Gena, C.: Enhancing cultural recommendations through social and linked open data. User Model. User Adap. Inter. **29**(1), 121–159 (2019). https://doi.org/10.1007/s11257-019-09225-8
30. Schmidhuber, J.: Optimal ordered problem solver. Mach. Learn. **54**(3), 211–254 (2004)
31. Trask, A., Hill, F., Reed, S., Rae, J., Dyer, C., Blunsom, P.: Neural arithmetic logic units. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS 2018. Curran Associates Inc., New York (2018)
32. Turing, A.M.: On computable numbers, with an application to the entscheidungsproblem. Proc. Lond. Math.Soc. **2**(1), 230–265 (1937)
33. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. arXiv:abs/1611.01578 (2016)