Dynamic and evolving Neural network as a basis for Artificial General Intelligence

Shimon Komarovsky[1][0000−0002−9036−0282]

Technion - Israel Institute of Technology, shiman@campus.technion.ac.il

Abstract. Artificial general intelligence (AGI) has to be founded on a suitable framework. Rule-based design is problematic, since it has to be manually updated if new and unaccounted for data is encountered. Current Deep Learning (DL) is also insufficient to become AGI, but it has the potential to be extended into one. Therefore an appropriate AGI has to be defined, followed by its appropriate DL implementation. We introduce an AGI, in the form of cognitive architecture, which is based on Global Workspace Theory (GWT). It consists of a supervisor, a working memory, specialized memory units, and processing units. Additional discussion about the uniqueness of the visual and the auditory sensory channels is conducted. Next, we introduce our DL module, which is dynamic, flexible, and evolving. It can be also considered as a Network Architecture Search (NAS) method. It is a spatial-temporal model, with a hierarchy of both features and tasks. Tasks can be objects, events, etc.

Keywords: Deep learning · General intelligence · Dynamic · Evolving.

1 Introduction

DL, as one of the Artificial intelligence (AI) approaches, is not as fully exploited as it could be. First, deep neural networks (DNNs) are passive models, since they have a fixed structure, while in reality there are dynamic processes, such as the neurons' construction/destruction in the brain. Second, "Learning" in DL is simply a categorization process without involving any thinking or imagination. Third, a successful DL model (DLM) requires its designers, to know the system. I.e. apply implicit or explicit prior knowledge in the DLM. Moreover, a carefully designed rule-based system may outperform a DLM, due to its dataset limitation, while a rule-based system is designed for much broader and more diverse scenarios. Fourth, DL is highly task-specific. Even multi-tasking in DL must be pre-defined. However real AGI can generalize not only on unseen data but also on unseen tasks (e.g. transfer/continual learning). Nevertheless, we propose a dynamic and flexible DLM that can be extended to AGI. We use DNN and not another machine-learning (ML) tool, since it is based on genuine intelligence.

2 Proposed architectures

In this section, we present an AGI architecture and a DLM, which can function as a module in this AGI architecture, e.g. in the perception/actuation module.

2.1 Proposed AGI model

A general AGI model sketch is shown in Fig. 1. This AGI is based on GWT [26, 29], which describes a multi-agent system. In GWT, the agents are local controllers that behave reactively, and compete with each other over access to the working memory (WM). Our AGI however, has no competition among its different and independent modules, i.e. processors and memories. Instead, it has centralized control with different elements, where each element has a specific function.

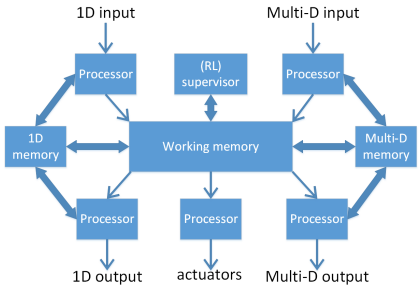


Fig. 1. AGI proposed architecture

Cognitive Architecture The diagram in Fig. 1 is also referred to as cognitive architecture, which represents an AI agent structure and functioning. For example, in the traffic control field we have two studies [12, 24] implementing AI which is based on distributed *cognitive* architectures [19].

The first study [24] is based on GWT, which contains agents imitating four brain function types: (i) sensory type, which holds positions and velocities of vehicles; (ii) behavioral type, which determines the light of the traffic signal in some intersection; (iii) consciousness type, which represents the WM and interacts with other brain functions; and (iv) motor type, which executes the chosen signal phase for each intersection.

In the second study [12], an AI is implemented on a single intersection using a Multipurpose Enhanced Cognitive Architecture (MECA) [11], which was adapted for the traffic signal control problem. The design consists of a Cognitive manager, a special kind of agent managing a set of physical objects made available on the Internet, providing information about themselves and receiving commands. E.g. it could be a car or an intersection agent. MECA is composed of two independent systems that communicate with each other: (i) a fast reactive system that holds the input sensors and output actuators, which is suited for normal situations, and (ii) a motivational system that is goal-oriented and suited for unexpected situations. It is thanks to automatic reactive and conscious elements, which together imply an intelligent behavior.

These papers are rule-based designs and they use small-scale networks. However, we strive for more adaptive and flexible designs, hence the use of DL instead of rule-based design. Additionally, the DL agent should be guided and supported by humans, as it occurs in infant-parent and student-teacher interactions.

Note that any cognitive architecture is limited and constrained by its structure. Therefore, we should have a wider view of it, thus considering the boundaries between the components to be not so well defined and perhaps changing.

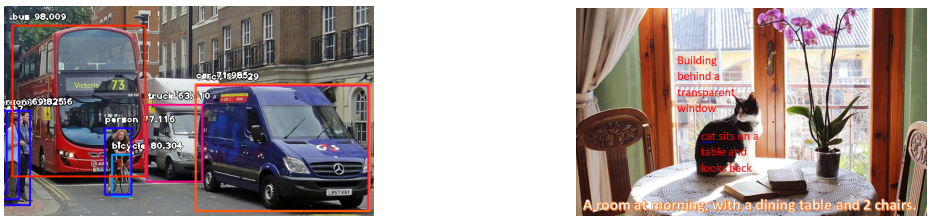
AGI functioning Just as in humans, we use 1D (audio) input and 3D (visual) input, but in contrast, we use those as outputs also. Moreover, we do not limit ourselves to the 3D spatial channel. I.e., more generally it can be 1 or more dimensions, dependent on the environment our agent is deployed in.

There is sequential processing of 1D and multi-D data separately, for feature extraction and categorization, of objects (static entities) or events (dynamic entities). Eventually, these objects propagate into the WM. In case a request or a need to produce some output by the agent occurs, such as in case of some idea or thought emerging, then the output can either go through the 1D channel, as humans describe their inner thoughts to the outer world verbally, or through a multi-D channel, and can be regarded as "screening imagination", which is like projecting the current thought into a screen. Because if we have two input types, a legitimate question arises: why should not there be outputs of these types also? Additionally, 1D information has a shared memory for both input, output, and the WM (such as language), denoted as 1D memory. Likewise it is in the multi-D information. The bidirectional arrows represent the acquisition (reading) and the update (writing) operations with the storage module.

The output communication of 1D and multi-D information can have various modes. It could be monitoring thoughts, or to wait for a meaningful output, or perhaps the AGI might have some sort of free-will to choose when to interact and via which of the two channels.

Notice, that this particular AGI is based upon Stimulus-Response behavioral theory [31], which claims that we cannot observe the mind itself, but rather only communicate with it. This assumption is similar to the Chinese Room Argument, since it means that we have no direct access to the operations within the agent, but only to its outputs. I.e., we have no explainability (black-box) over the intelligence inner operations. We have only the output, which we can analyze. It is also called "intelligent behavior", and it is expressed via human productivity over history, such as sciences, psychology, technology, and more.

This AGI does more than merely static objects identification, as it is in DL. It extends to the temporal dimension, where it is not only about objects, but also about events, objects' behavior and function, associations from past experiences, and more. It is illustrated in comparison between current AI and AGI, in Fig. 2.



(a) AI identifying. (b) AGI interpretating.

Fig. 2. Comparison between AI and AGI comprehending the environment.

Two information types Given that we permit a multi-D output in our AGI, we can ask: why do not humans have imagery tool as output? One can argue it would hurt our basic desire for privacy, but then just as we choose whether to talk or not, we can similarly choose when to turn this tool on.

Our current opinion is that the world we see with our eyes is what we all agree upon. Other than that, our inner models of the world are totally different.

And why do not we have a symbolic or linguistic channel to be objective as vision? Why do we end up with inner and unique symbolic representation? We think it is because language is highly context-dependent, and since each person has different contexts along with his life, or different experiences, then he develops a different meaning/feeling/understanding of the objective concepts we all agree upon. Hence, the concepts we use in external communication are objective and common to all people, but their interpretation is different for each one.

Therefore physical reality function is only for the objective agreement for effective communication between us, which is realized via language. However, vision is not a communicative channel for us.

Consequently, the purpose of having two channel types is for distinguishing the outer and inner world that the agent interacts with. Furthermore, humans (as should be followed by AGI) base their inner representation on spatio-temporal events, or operational language (i.e. objects and actions/attributes), which can be expressed by words or any symbolic language. Therefore spatio-temporal information can be transferred to humans not by the objective world (static visual information), but rather by language, which in our case is in a 1D sequential form. Agents denoted as green circles, communicating via 1D and individually perceiving multi-D input are illustrated in Fig. 3.

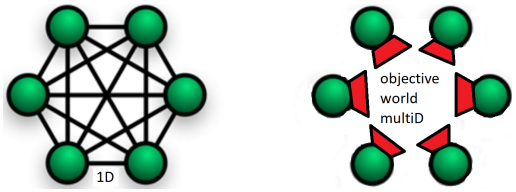


Fig. 3. Objective (right) verse Subjective inner representation (left).

Notice, that supervised DL is actually going around intelligence development. Since an infant starts from unsupervised learning, and only after a long period can make the connection between the objects he has seen/heard so far and their meaning. We continue this line of thought in the next subsection.

2.2 Proposed DLM

Until now we presented a general AGI model. Now we turn to discuss which DLM can implement such AGI, or implement each or some of its different modules.

Some necessary background is needed. We start by reviewing prior knowledge in DL, also known as inductive bias. Then due to difficulties with matching the most proper prior knowledge to each specific problem we encounter - we continue with reviewing NAS that attempts to deal with these difficulties. Finally, we present our DLM as another possible NAS strategy.

Prior knowledge in DNNs As shown from studies in the traffic signal control field [21, 33, 34], CNN and RNN present better prediction results in accuracy and stability compared to other ML methods. However, this might be due to them being tailored to their particular problem. Apparently, CNN and RNN are examples of simplification of the general NN, since they are more intuitive and understandable, with the price of being task-specific [33].

However, these special structures have fewer variables and contain more prior knowledge, compared to fully-connected (FC) layers of regular (vanilla) NN. Hence we can see the transformation from FC to CNN or RNN as localization, where the network structure is designed specifically for the data it handles.

Prior knowledge can appear in many forms: in the structure general type, in the hyper-parameters, in the regularization method (e.g. dropout, constraints, etc), in the decision about sharing or grouping [28] or separating features/variables and more. E.g. when we set apart traffic data from weather data [17], or roads from stations [15] and fuse them only later. We can also separate tasks into groups [15]. [16] suggests sparsifying the NN (e.g. removing connections in CNN, or grouping/sharing parameters), which results in fewer parameters and more prior knowledge and efficiency. It also removes redundancy.

However, all these structures can be too restricted or best perform for narrow data variations. Hence many studies try different hyper-parameters or architectures, to get better performance. E.g., they use Network Architecture Search.

Network Architecture Search (NAS) In NAS, e.g. AutoML [2], we first define/restrict the search space for models. Usually, reinforcement learning (RL) or Evolutionary Algorithms (EA) are used as search techniques. In hyper-parameters tuning often grid and random search are used.

A more efficient (faster) approach use weight sharing between proposed networks, instead of putting them in the search and training them from scratch (such as in EA). They do it by sampling sub-networks from a large parent network where they force all sub-networks to share weights. Only the best final model is trained from scratch. Alternatively, we can modify only some specific parts in NN, while leaving the rest unchanged.

Unlike previous EA and RL, in Differentiable Architecture Search (DARTS) [20], the search is defined as a differentiable and continuous problem. DARTS have some unknowns so they use multiple categories, where the final architecture is discretized after the search is over.

Hypernetwork [8], is where the weights are learned not by training original NN but rather determined via other NN. I.e., for every tested input, there will be different weights to predict the output. E.g., in [13] there is a hypernetwork

for both CNN and RNN as two ends of a spectrum, which allow it to be a relaxed weight-sharing approach and allows us to control the trade-off between the number of model parameters and model expressiveness. In our opinion, Hypernetworks can be extended by constructing also a second network to learn the weights of the first one that learns weights for the main one. And so forth, strengthening the evolution in different time resolutions/scales (from slow to fast).

When we use EA for NAS, it is referred to as Neuroevolution [9]. Beyond generating DNNs, it is also used in generating hyperparameters, NN building blocks, activation functions, and even the algorithms for learning (rules). E.g. in NEAT [6], it is used to replace back-propagation with Genetic Algorithm (GA) or combine them both, in searching for weights.

In the SMASH method [3], they train a hypernetwork to predict a network weights in one shot instead of training all those candidate networks from scratch. They first train a hypernetwork to predict the weights of some given arbitrary network, then they randomly generate many architectures, and rather than training those models, they use a hypernetwork to obtain the weights. Finally, they pick best performing architecture and train it from scratch.

Detailed proposed DLM Since we live in spatial-temporal reality, the AGI should be processing spatial-temporal information. Therefore, our DLM tackles a time-sequential visual input. Next, we describe its evolution through the eyes of a newborn infant. Similar models to our DLM can be found in [25].

We presume, that an infant does not have any supervised learning at the beginning of his life, rather an unsupervised one. The first thing he does is segment the time period into simple events. But he starts with a single event detection (e.g. his total awaking period) through some initial DNN. See Fig. 4(a).

After a while, when enough counts detected the single event, a split performed of this event into two (or more) classes of events, e.g. day and night, see Fig. 4(b). By counts we mean the number of times the output class was triggered. Now it can differentiate two events, sharing the same deep features. Later he can extend the number of events, and recognize as many events as necessary. Hence, it is an adaptive NN structure, adaptive by necessity.

At some point of evolution, when connections (weights in DNN) and event identification (output layer's counts) is strengthened and established, the model can change its attention (or free its resources, since the given level became more automatic, similar to the idea in [14]). It can build a new layer or level if a simultaneous re-occurrence of several events is detected. E.g. the re-occurrence of seeing the mom appearing and preparing herself to give milk suggests to the infant that it is a composite event on its own, see Fig. 4(c).

Opposite structure-changing operations could be (i) deleting extremely rare nodes/edges in the DNN, a bit similar to dropout regularization in DL; and (ii) decomposing an event, if it appears to be more complex than it was supposed to be. I.e., turn it to be fine-grained. In other words, if previously it was treated as a specific-level event, now it is decomposed into simpler events (refinement).

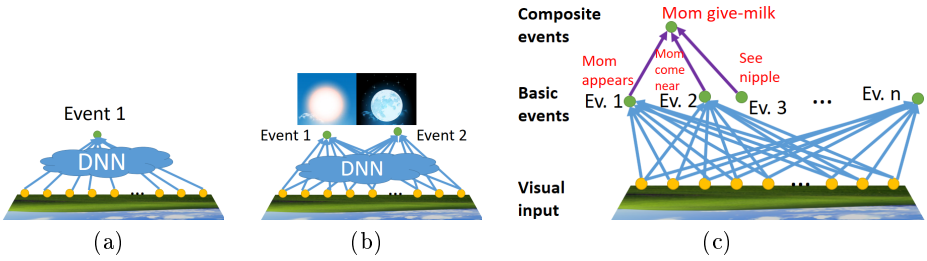


Fig. 4. Neuron separation and composition in the proposed approach (Ev.=Event).

It is decomposed into either existing events or new ones. If new ones, then they have to be attached to lower-level events/features. E.g. see Fig. 5, where we decompose the "ball in the air" event into its three basics.

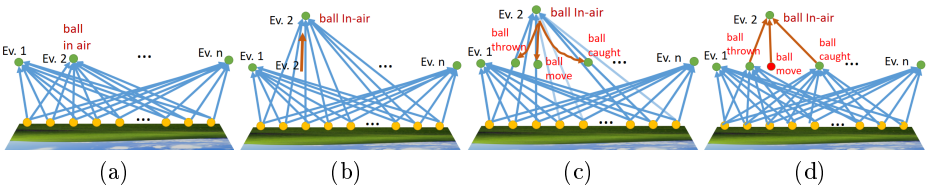


Fig. 5. Neuron decomposition in the proposed approach (Ev.=Event).

It does not have to be risen up to the higher layer. Rather it can stay at the same level and be decomposed into lower levels. Both options may be available for reconstruction operators, because sometimes it is better to stay at the same level, and sometimes it is better to level up.

Suggestion: In decomposition, we create new events anyway, because we do not know which of them are really new or exist already. Therefore, if two or more events from the same layer, occur simultaneously very often, then it might imply that they represent the same event, hence should be reduced to one.

This approach is self-supervised and not unsupervised, since we do not perform clustering into a pre-defined number of categories. Here, the number of categories, connections, etc - are all dynamic, similar to NAS, and changed by the decision of some supervising algorithm.

Another reason for this dynamic algorithm is that real intelligence does not end up with categories like a cat, dog, etc. Moreover, most AI research, either visual or textual, works backwards. It always start from high-complexity data and try to reach it. Instead of simple to complex method as it should be. Therefore AGI has to be evolving.

For this dynamic algorithm to work, we have to store the number of visits of each weight (edge in DNN) and of each node in the DNN. If scalability is an

issue for large DNNs, we can reduce the visits memorization from being stored for each neuron to being stored in each cluster of neurons in large enough DNN.

Furthermore, the visits can occur during “waking" periods (when the DNN is fixed), and the structure update can be done during “sleeping" periods, when there is no stimulus from the sensors, while the trajectory frequency within the DNN is stored in the neurons themselves, as mentioned above. Perhaps it may even be so in the actual human brain. The rate of changing structure can also be modeled with a learning rate as in RL, where at first is mostly exploration (i.e. fast NN growth), and then lesser exploration and more exploitation. Finally, we can assume a finite number of nodes and connections, i.e. limited resources (so that it will not grow infinitely), and adjust the learning rate accordingly.

Another issue can be in recoverability: what if in the future we would regret some of the elements that were deleted from the DNN. Can they be restored? One way to deal with it, is by the assumption that the infant first should grow up, and refine its categorization, i.e. we should start with only enlarging the network. Only in a big enough network, we can start deleting nodes/connections, because then the agent accumulated enough confidence/experience.

It is important to note that in the common DNN, the structure is supposed to contain features in neurons, but these features depend on the data. Hence the necessity of checking several structures, as in NAS: since we are probably wrong about guessing the number of best-describing features at each layer. Dynamic NN deals with this issue, by keeping only the relevant and true features.

One issue in dynamic structure NN could be splitting or deleting features that are supposed to be frequent or rare, yet we should leave them as they are. One way to deal with it is via relative frequency, i.e. to have some min-max range of relative frequency among neurons (all of them or for each level or something else), which will not be split/deleted. Only those extremely frequent/rare outside of this range would be split/deleted.

This dynamic algorithm is less computational since it has fewer connections compared to FC NN, similar to sparse NN. Some optional additions to the DLM could be:

* The task/output is an event, but it could be any object or action, etc.
* We can probably update also the inner neurons, and not only the outer ones.
* Allow reallocation of activated-together neurons, such that they could be in proximity, which can ease on computation and improve ordering.
* We should constrain the model in all its adaptive parameters, e.g. the number of total layers, number of neurons per layer, and so on. It is due to our computation limitations (in time and storage), and to regularize over the evolution. I.e., eliminate redundancy and encourage competition/trade-offs. E.g., the learning rate and the new neurons generation will be scaled accordingly to the given limits.

In addition, the brain plasticity principle and the task distribution over many regions in the brain, may imply that neuronal information transmission (chemical and electrical) is not the only operation type that occurs in an active human

brain. I.e., similarly to cell division, reproduction, and then assignment to different roles stated by the genome, it may give neurons different and perhaps adapting roles. That is, besides the adaptation and reconstruction of NN architecture, the neurons themselves can change their function, e.g. become convolutional, recurrent, recursive, attentive, etc.

Task hierarchy in the proposed DLM Note that the extending of classes converts this DLM into hierarchical multi-class DNN. Such DNNs exist in the literature. In [4] we have feature layers and then task layers. Sometimes these layers can be mixed up. Some find labels structure separately from the model [5, 23], and some do it as a part of the model [10, 18, 22]. Some put all the tasks to be learned over one classifier, i.e. globally or in the last layer of the NN [5,30], and some insert intermediate tasks inside the NN, i.e. locally [4, 22, 27, 32]. Some learn the structure from the data [18], e.g. by unsupervised clustering of the labels via some similarity measure [23], and some import the structure from external knowledge base [7], or use it to adapt this structure [5, 10].

Important note: unlike features that are distributive representatives of data, which hold only some piece of the actual information, tasks are end-point independent representatives of data, thus ruining in a way, the distributive nature of NN. However, this is not their main drawback. The fact that they are informational points - enforces a huge memory, since we need lots of them to represent a huge amount of terms and concepts. As opposed to a small group of inter-related features, which can characterize an enormous amount of input data. Therefore, at some point, we should consider replacing/converting tasks into features.

Generally, there may be different hierarchies besides compositional ones, e.g.: family tree, parts of speech, table of contents, topics and sub-topics, etc. We propose the evolution to develop into different hierarchies, just like tree expansion: in different locations of a given NN and in different structures. An illustration of multiple hierarchies formed in a given NN is in Fig. 6.

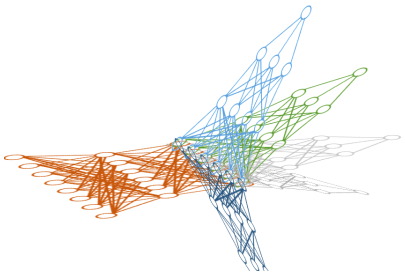
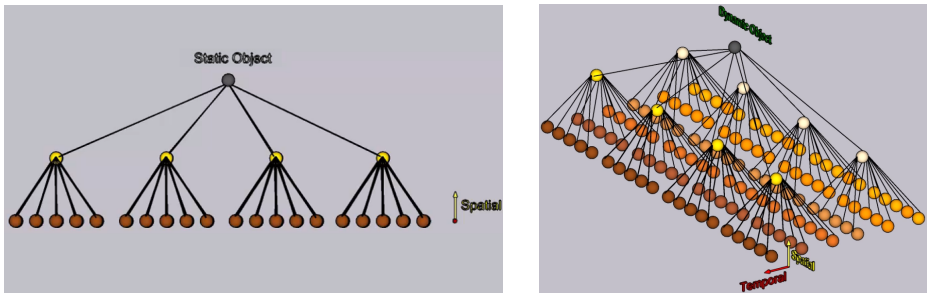


Fig. 6. Branching due to multiple hierarchies.

Temporal dimension of the DLM Until now the DNNs we have seen in the proposed DLM use regular static structure, i.e. a single simultaneous set of inputs produce a single simultaneous set of outputs, in other words, we have not seen any recurrent connections to include a temporal sequence of inputs.

Usually, spatial-temporal models combine CNN with RNN in different ways. Either separately: CNN→RNN or interchangeably: CNN→RNN→CNN→RNN... Another way: separate them (CNN and RNN) for separate inputs, e.g. textual for RNN and visual for CNN, with a fusion module at the end. Hence, event tasks, such as classification/clustering, can be done using the methods above.

On the other hand, we know that regular FC DNNs, are used for spatial object tasks. But If we want to extract features along the temporal dimension also, we can simply add recurrent connections. Alternatively, we can extend the DNN to include the temporal dimension, without changing the spatial dimension, i.e. orthogonal to it. See Fig. 7 with static and dynamic object tasks.



(a) (Front view) Objects tasks (b) (Side view) Events tasks

Fig. 7. Spatio-temporal DNN model.

In Fig. 7 is shown a FC NN. However, if we want - we can specialize it in different ways, e.g. shared connections/parameters or convolutions, etc. And we can do it for either the spatial or temporal dimensions, or both.

3 Final words

In our opinion, AGI's main purpose is to organize information to be utilized optimally in a variety of tasks. Hence, prediction might be only a tool to estimate this main goal. We also believe that DNN is an efficient algorithm and memory structure, which can achieve this goal, since it organizes the data with the intention to have a fast recovery of this data, later.

We advocate that efficiency is more important than effectiveness, in AGI, since it is about the exploitation of available resources, while effectiveness is about how well a goal is achieved [1]. E.g. the common attitude in DL to compare performances.

Other characteristics an AGI should have are those that imitate humans: correct teaching order (simple to complex), ability to grow/evolve in compulsory stages, and more.

References

1. https://simania.co.il/bookdetails.php?item_id=145306
2. AutoML: Automl, https://towardsdatascience.com/how-to-apply-continual-learning-to-your-machine-learning-models-4754adcd7f7f
3. Brock, A., Lim, T., Ritchie, J.M., Weston, N.: Smash: one-shot model architecture search through hypernetworks. arXiv preprint arXiv:1708.05344 (2017)
4. Cerri, R., Barros, R.C., De Carvalho, A.C.: Hierarchical multi-label classification using local neural networks. Journal of Computer and System Sciences **80**(1), 39–56 (2014)
5. Chen, H., Miao, S., Xu, D., Hager, G.D., Harrison, A.P.: Deep hierarchical multi-label classification of chest x-ray images. In: International Conference on Medical Imaging with Deep Learning. pp. 109–120. PMLR (2019)
6. Chen, L., Alahakoon, D.: Neuroevolution of augmenting topologies with learning for data classification. In: 2006 International Conference on Information and Automation. pp. 367–371. IEEE (2006)
7. Feng, S., Zhao, C., Fu, P.: A deep neural network based hierarchical multi-label classification method. Review of Scientific Instruments **91**(2), 024103 (2020)
8. Galanti, T., Wolf, L.: On the modularity of hypernetworks. Advances in Neural Information Processing Systems **33**, 10409–10419 (2020)
9. Galván, E., Mooney, P.: Neuroevolution in deep neural networks: Current trends and future challenges. IEEE Transactions on Artificial Intelligence **2**(6), 476–493 (2021)
10. Gu, J., Zhao, H., Lin, Z., Li, S., Cai, J., Ling, M.: Scene graph generation with external knowledge and image reconstruction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1969–1978 (2019)
11. Gudwin, R., Paraense, A., de Paula, S.M., Fróes, E., Gibaut, W., Castro, E., Figueiredo, V., Raizer, K.: The multipurpose enhanced cognitive architecture (meca). Biologically Inspired Cognitive Architectures **22**, 20–34 (2017)
12. Gudwin, R., Paraense, A., de Paula, S.M., Fróes, E., Gibaut, W., Castro, E., Figueiredo, V., Raizer, K.: An urban traffic controller using the meca cognitive architecture. Biologically inspired cognitive architectures **26**, 41–54 (2018)
13. Ha, D., Dai, A.M., Le, Q.V.: Hypernetworks. CoRR **abs/1609.09106** (2016), http://arxiv.org/abs/1609.09106
14. Hawkins, J., Blakeslee, S.: On intelligence: How a new understanding of the brain will lead to the creation of truly intelligent machines. Macmillan (2007)
15. Huang, W., Song, G., Hong, H., Xie, K.: Deep architecture for traffic flow prediction: deep belief networks with multitask learning. IEEE Transactions on Intelligent Transportation Systems **15**(5), 2191–2201 (2014)
16. Ioannou, Y.A.: Structural priors in deep neural networks. Ph.D. thesis, University of Cambridge (2018)
17. Koesdwiady, A., Soua, R., Karray, F.: Improving traffic flow prediction with weather information in connected cars: A deep learning approach. IEEE Transactions on Vehicular Technology **65**(12), 9508–9517 (2016)
18. Li, Y., Ouyang, W., Zhou, B., Wang, K., Wang, X.: Scene graph generation from objects, phrases and region captions. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1261–1270 (2017)
19. Lieto, A., Bhatt, M., Oltramari, A., Vernon, D.: The role of cognitive architectures in general artificial intelligence (2018)

20. Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055 (2018)
21. Ma, X., Tao, Z., Wang, Y., Yu, H., Wang, Y.: Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Transportation Research Part C: Emerging Technologies **54**, 187–197 (2015)
22. Nguyen, D.K., Okatani, T.: Multi-task learning of hierarchical vision-language representation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 10492–10501 (2019)
23. Papanikolaou, Y., Tsoumakas, G., Katakis, I.: Hierarchical partitioning of the output space in multi-label data. Data & Knowledge Engineering **116**, 42–60 (2018)
24. Paraense, A.L.O., Raizer, K., Gudwin, R.R.: A machine consciousness approach to urban traffic control. Biologically Inspired Cognitive Architectures **15**, 61–73 (2016)
25. Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., Wermter, S.: Continual lifelong learning with neural networks: A review. Neural Networks **113**, 54–71 (2019)
26. Reggia, J.A.: The rise of machine consciousness: Studying consciousness with computational models. Neural Networks **44**, 112–131 (2013)
27. Sanh, V., Wolf, T., Ruder, S.: A hierarchical multi-task approach for learning embeddings from semantic tasks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 6949–6956 (2019)
28. Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: European Semantic Web Conference. pp. 593–607. Springer (2018)
29. da Silva, R.C.M., Gudwin, R.R.: An introductory experiment with a consciousbased autonomous vehicle. In: 4th Workshop in Applied Robotics and Automation (2010)
30. Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., Xu, W.: Cnn-rnn: A unified framework for multi-label image classification. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2285–2294 (2016)
31. Watson, J.B., Meazzini, P.: John B. Watson. Il mulino (1977)
32. Wehrmann, J., Cerri, R., Barros, R.: Hierarchical multi-label classification networks. In: International Conference on Machine Learning. pp. 5075–5084 (2018)
33. Wu, Y., Tan, H., Qin, L., Ran, B., Jiang, Z.: A hybrid deep learning based traffic flow prediction method and its understanding. Transportation Research Part C: Emerging Technologies **90**, 166–180 (2018)
34. Yu, H., Wu, Z., Wang, S., Wang, Y., Ma, X.: Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks. Sensors **17**(7), 1501 (2017)