The Spiral Discovery Network as an Evolutionary Model for Gradient-Free Non-Convex Optimization

Adam B. Csapo
Széchenyi István University
Dept. of Informatics
Győr, Hungary
csapo.adam@sze.hu

Abstract—The Spiral Discovery Method (SDM) was originally designed as a cognitive artifact to help manage the complexities of manually tuning parametric models in high-dimensional parametric spaces. Recently, several modifications and enhancements were proposed to SDM with the goal of making it suitable for tasks requiring automated non-convex optimization besides manual parameter configuration. The key challenge behind such enhancements – collectively referred to as Spiral Discovery Network models (SDNs) based on their network-based formulation – is how to replace human intuition in maintaining the adaptivity of the search process. In this paper, recent advances behind SDN models are summarized, and their theory is further developed with the goal of making them useful for the optimization of multi-level, hierarchical architectures. Results from experiments directed at optimizing convolutional networks on the MNIST dataset are presented in order to highlight the strengths and weaknesses of the approach.

Index Terms—Gradient-free optimization, Non-convex optimization, Spiral Discovery Method, Spiral Discovery Network

I. Introduction

The debate on whether evolutionary methods or neural networks are better suited to tackle problems in artificial intelligence is strongly related to the classical debate on nature versus nurture [9]. Clearly, both classes of approach are important components of human intelligence, and both have their pros and cons from a computational point of view. Evolutionary methods are well suited to finding good candidate solutions in large parametric spaces, however they often rely on the availability of hand-coded genotypes that are expected to yield useful results following the classical operations of recombination and genetic mutation – a constraint that can be considered as arbitrary and limiting given that 1.) different encodings will lead to different solutions, and 2.) the classical genetic operations may not always be well suited to the problem domain. At the same time, (unsupervised) neural networks are known to be capable of automatically generating high-dimensional input encodings that are often useful for the problem at hand, thus alleviating the need for hand-coded inputs; however, their applicability rests on the assumption that their performance can be evaluated using a loss function that is effectively computable and differentiable.

In cases where the loss function associated with a problem is unknown or difficult to compute and / or differentiate, the problem of assigning credit to the many individual components involved in the functionality of a neural network becomes intractable (for more on the credit assignment problem see e.g. [19], [15]). Cases like these might include:

- Loss functions that are non-differentiable, e.g. because they are black box models or rely on human feedback;
- Loss functions that are undefined for many inputs (e.g. loss functions that involve a ranking of different configurations, or a selection of the top n configurations out of many

In such cases, methods based on coarser-grained feedback, such as reinforcement learning or direct search in weight space are often used [19].

A recently proposed class of search models – referred to as Spiral Discovery Network (SDN) models – can be considered as an alternative approach to all of the above mentioned methods. SDN models have the following properties:

- Instead of relying on evolutionary operations such as recombination and mutation, SDNs operate directly within the search space, and so the requirement of creating useful, hand-coded input encodings is alleviated;
- To compensate for the lack of (genetically-motivated) operations for generating new candidate solutions, SDNs auto-regressively explore the search space along a parametric hyper-spiral structure;
- The application of this hyper-spiral structure in turn implicitly generates differential feedback information (besides the coarse-grained feedback obtained via each candidate solution), allowing the model to adapt its behavior in subsequent cycles of exploration.
- Difficulties arising from the search being conducted directly in the high-dimensional parameter space can be alleviated somewhat by using hierarchical SDN models based on a partitioning of the search space – as described later in this paper.

In this paper, the origins of SDN models are presented, as well as a summary of recent advances in their formula-

978-1-5386-7094-1/18/$31.00 ©2018 IEEE

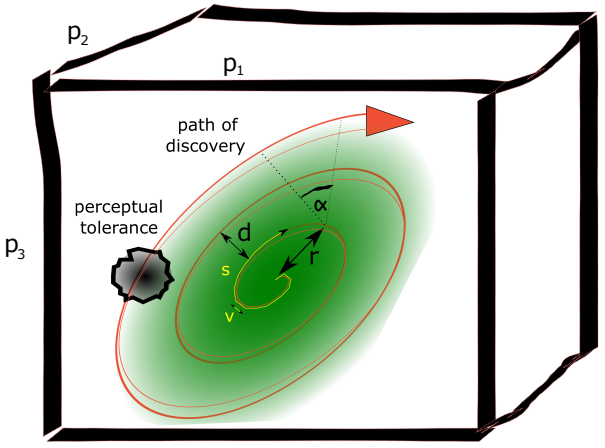

Fig. 1. Conceptualization of SDM. The key parameters of the hyper-spiral (denoted by r, d and α) along which the parameter search is to be conducted can be set based on already known, sub-optimal candidate solutions, so that users need only specify the distance and velocity ($\equiv$ step size) along the hyper-spiral.

tion and applicability. Two ways of introducing hierarchy – and thus improving computational tractability - are considered, including a newly developed proposed approach using parameter space partitioning. The applicability of the latest hierarchical model is then exemplified through an experimental design involving the training of a classical convolutional neural network on the MNIST dataset.

The paper is structured as follows. Section II briefly recapitulates the motivations behind and the design of the original Spiral Discovery Method as well as the latest advances in the formulation of Spiral Discovery Networks. Finally, sections III and IV are dedicated to newly proposed enhancements and their validation on the problem of training a convolutional neural network to recognize hand-written digits.

II. Motivations and Background behind SDM and the SDN Framework

A. The Spiral Discovery Method (SDM)

The original motivation behind the spiral discovery framework came from the problem domain of designing useful multimodal user interfaces [1], [5]. In applications where the performance of a system has to be tuned based on user feedback, it is difficult to obtain feedback that is both consistent and available at a high input resolution.

The key idea behind SDM is to systematically reduce the number of tuning parameters presented to users at any given time. The two qualifications in the last sentence ("systematically", and "at any given time") are important. The latter means that although users have limited options, the semantics of those options (i.e. the principal component of the search direction) are always changing. The former, in turn, means that although the semantics of the tuning options change all the time, they change in predictable ways, which allow users to effectively refine the direction in which they subsequently wish to continue the search.

The conceptual model behind the original SDM is shown in Figure 1. It is worth noting that the reliance of SDM on continuous user feedback to facilitate optimization makes it in some sense comparable to the framework of interactive evolutionary computation [22], [23], as well as interactive deep reinforcement learning [8]. Note also that from a broader CogInfoCom perspective, solutions that rely to some degree on the concept of "humans in the loop", or otherwise consider human cognitive capabilities towards the interpretation of digital content can all serve as inspiration. For example, at a very high level, the research area of mathability is strongly relevant to the human-in-the-loop concept [2], [3], [4]. Similarly, research areas including cyber-learning [11], [13], [12], representation of uncertain knowledge [17], [16] and cognitive visualization [24], [20] all have something to say about how human cognitive representations can be used to support a wide range of CogInfoCom applications.

B. The Spiral Discovery Network Framework

The motivations behind the development of Spiral Discovery Networks was two-fold. First, because SDM offers new insights into the problem of non-convex gradient-free parametric search in general, the question naturally arises whether the need for human interventions (and human feedback) can be taken out of the picture, leading to a more generally applicable search tool. Second, in a technical sense it can be expected that a re-formulation of the original SDM in terms of recurrent neural network architectures may help bring to light useful properties of the model – given that the theory and terminology behind neural networks are already well developed and well-suited to discussing optimization problems.

In broad terms, the network-based formulation of SDMs – referred to as the Spiral Discovery Network (SDN) framework – consists of the following components [6], [7]:

- A principal component that guides the primary search direction in each cycle
- A timer module that functions as a modulo counter for updating the state of the cell at discrete time steps within each cycle
- A perturbation module that determines the direction in which, and the extent to which the slope of exploration is to be modified at each time step within each cycle
- A hypervisor module that refreshes the hyperparameters of the perturbation module based on feedback signals obtained – not necessarily in synchrony with the timer module. (While the latter qualification did not apply to earlier formulations, the parameter space partitioning based hierarchical SDN proposed in this paper makes use of this added property.)

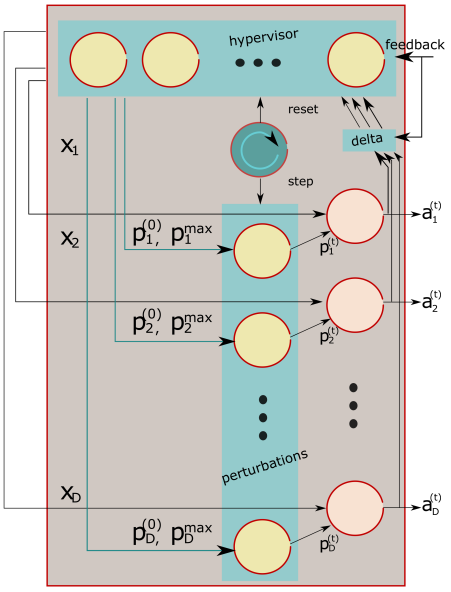

Fig. 2. Neural network inspired formulation of an SDN cell.

- A delta module that models the influence of changes in each of the output dimensions on the value of the loss function

A graphical representation of an SDN cell and its components is shown in Figure 2. The updated activation at time t is:

$$\mathbf{a}^{(t)} = (t * step\_sz + 1)\mathbf{x} + \mathbf{p}^{(t)} \tag{1}$$

where:

$$\mathbf{p}^{(t)} = \mathbf{p}^{(t-1)} + sgn(cycle\_dir^{(t)}) \cdot \frac{\mathbf{p}^{max} - \mathbf{p}^{(0)}}{\frac{cycle\_len}{4}}$$
$$cycle\_dir^{(t)} = \begin{cases} 1 & , \text{if } \frac{cycle\_len}{4} \leq t < \frac{3*cycle\_len}{4} \\ -1 & \text{otherwise} \end{cases} \tag{2}$$

Generally speaking, the state of the SDN cell is updated in a series of timesteps which together constitute optimization cycles. In the update equations, $\mathbf{x}$ refers to the (normalized) principal component vector – the general direction in the parametric space that is being explored by the cell, while $\mathbf{p}$ refers to the perturbation vector that is added to the principal component. The relationship between the two is governed by the hyperparameter *step_sz*, which ensures that the path of parametric discovery expands in the general direction of the principal component (note that *step_sz* represents the degree of exploitation in the optimization process). The direction and norm of $\mathbf{p}^{(t)}$, by contrast, which ultimately depends on the relationship between $\mathbf{p}^{(0)}$ and $\mathbf{p}^{(max)}$, determines how far from the principal component the exploration will deviate (therefore, it is directly related to the concept of degree of exploration in the optimization process). *cycle_dir* governs the direction in which the perturbations are changed, and is dependent on the length of the cycle as well as the current phase within the cycle. The values of $\mathbf{p}^{(0)}$, $\mathbf{p}^{max}$ and $\mathbf{x}$ are dependent on the cycle (or more precisely, on the discoveries made during the previous cycle), and are initialized as follows:

$$\begin{aligned} p^{(0)}_{i,unnormed}[c] &= p_i^{(\arg\min_t h_i^t[c-1])}[c-1] \\ p^{max}_{i,unnormed}[c] &= p_i^{(0)}[c] + \\ &+ softmax(\sigma_{h_i}[c-1])(\sigma_{h_i}[c-1]+1) = \\ &= p_i^{(0)}[c] + \frac{\exp\sigma_{h_i}[c-1]}{\sum_I \exp\sigma_{h_I}[c-1]}\left[\sigma_{h_i}[c-1+1]\right] \\ \mathbf{p}^{(0)}[c] &= ||\mathbf{p}^{(0)}_{unnormed}[c]|| \\ \mathbf{p}^{max}[c] &= ||\mathbf{p}^{max}_{unnormed}[c]|| \\ x_i[c] &= ||x_i[c-1] + \frac{p^{(0)}_{i,unnormed}[c]}{||\mathbf{p}^{(0)}[c]||}|| \end{aligned} \tag{3}$$

Here, the value of a parameter within a cycle c is represented using square brackets, so that for example $h_i^t[c-1]$ refers to the value of the i-th hypervisor cell at time t of cycle $c-1$. σ_{h_i} denotes the standard deviation of value the i-th hypervisor cell. The update equations ensure that:

- the perturbations in the new cycle are centered, in each dimension, around the perturbation that was associated with the lowest cost function value in the previous cycle (note that h_i refers to the i-th hypervisor cell)
- the maximum values of the perturbations are set to their starting value, plus a value that depends on the standard deviation of the corresponding hypervisor cell in the previous cycle, as well as its relation to the standard deviations of other hypervisor cells. In general, the larger the deviation in a given dimension in an absolute sense, the greater the distance will be between the initial and maximal perturbation in the following cycle (in which case the network will be more explorative in that dimension). Similarly, exploration in dimensions that are characterized by large relative deviations will also be higher in the following cycle.
- the principal component, $\mathbf{x}$ is set to the initial principal component plus the normalized value of the initial perturbation (which, of course, is a scaled version of the best perturbation in the previous cycle).

Finally, the specification of hypervisor cells is as follows. Given the fact that the SDN model makes no assumptions on the loss function, it cannot be assumed that this influence can be modeled effectively over long periods of time, or even that it can be modeled using a differentiable function. Therefore, changes through longer periods of time are discounted based on the following update equation to the hypervisor cells:

$$h_i^{(t)} = loss^{(t)} * \gamma_{h_i}^{(t)} = \\ = loss^{(t)} * softmax\left(\sum_{d=1}^{\infty} d^{-1}(a_i^{(t)} - a_i^{(t-d)})^2\right) \quad (4)$$

where γ is a scaling factor applied to the loss function value that is calculated through a softmax function and (as a result of the softmax function) depends on all dimensions of the output activity. The scaling factor γ ensures that those losses are most taken into consideration that are accompanied by relatively large changes in output. Since this is computed separately in each dimension of the output, the hypervisor activation in each dimension will be most influenced by loss values that occur when it is mostly the activation in that dimension that is changed.

III. Newly Proposed Enhancements

The first enhancements to the Spiral Discovery Network framework were proposed in [7]. In that work, the idea was presented that SDNs can be turned into hierarchical models by adding a separate SDN for setting the hyperparameters of the search, and it was shown that this enhancement alone allows for optima with the same – or better – loss values to be found more quickly.

In the current paper, a more generally applicable approach is proposed, which consists of breaking up the search space into disjoint partitions, and using separate SDNs to search within those partitions. The key challenge then becomes how to integrate the outputs of each of the SDNs, and how to assign credit to various configurations explored by them as a function of a single global loss value. This challenge is further amplified when the so-called temporal slowness principle is used – a principle which applies to the human brain, and which has been found to be essential when dealing with complexity (for a few examples, see e.g. [10], [21], [25]).

To address this challenge, three factors have to be taken into consideration (see also Figure 3):

- If multiple SDNs are operated with different cycle lengths, their output activations have to be up- or down-sampled to enable the generation of as many global output activations as required by the application (top row in Figure 3)
- If multiple SDNs are operated with different cycle lengths and a single feedback signal, the resolution of the feedback signal needs to be up- or down-sampled, or otherwise aggregated to match the cycle length in each of the SDNs (top row in Figure 3)
- If multiple SDNs are operated with different cycle refresh rates, SDNs that produce repeated cycles have to take into consideration all the feedback signals for each step of the cycle when updating their hypervisors (second row, right-hand side in Figure 3)

A relatively simple form of aggregation (i.e., addition) is used in Figure 3 to solve these three issues.

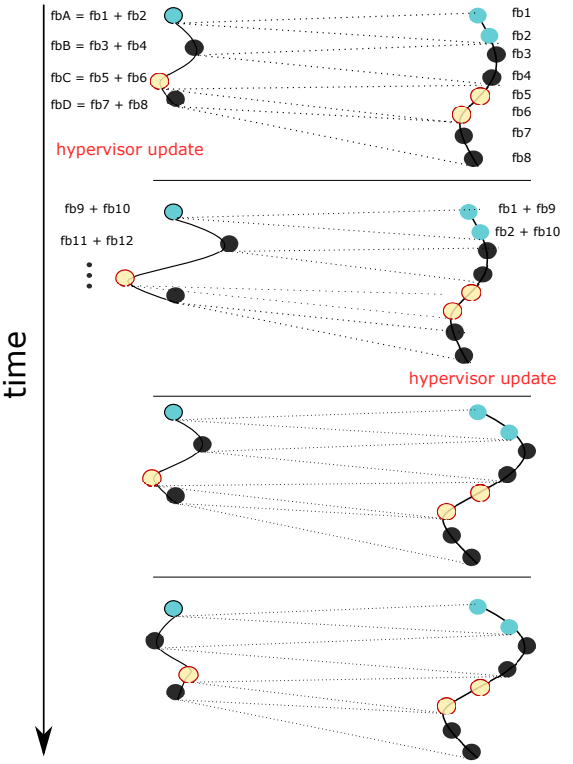

Fig. 3. Conceptual diagram on the synchronization of 2 SDNs with different cycle refresh rates and different cycle lengths. In the figure, *fbX* stands for the x-th feedback. The SDN on the left produces cycles at a higher refresh rate, but each of the cycles are shorter than those produced by the SDN on the right. The proposed enhancment to SDNs consists of partitioning the dimensions of the high-dimensional parameter space, allocating the partitions to different SDNs and up-sampling the activations of those SDNs which produce shorter cycles to obtain feedback information that is then evenly aggregated among the available parameter configurations. In this figure, the first cycle of the SDN on the left is shorter than that of the one one on the right, thus for each output activation produced on the left, the effects of two output activations on the right (in terms of two different global loss values, such as fb1 and fb2, or fb3 and fb4) are aggregated and used to update the hypervisors on the left (whereas a single loss value is attributed to each activation on the right). At the same time, because the refresh rate for the cycles withinin the SDN on the right hand side is smaller, feedbacks obtained for the same step of the same cycle need to be aggregated prior to updating the hypervisors.

IV. Experiment: Training Convolutional Neural Network to Recognize Handwritten Digits

The MNIST database is a benchmarking tool that contains a training set of 60,000 and a test set of 10,000 low-resolution, handwritten digits [14]. The goal of the experiment presented in this paper was to compare the performance of parameter configurations obtained through a classical gradient-based training process and a hierarchical SDN based approach.

The experiment was conducted on a convolutional neural network implemented using the PyTorch library

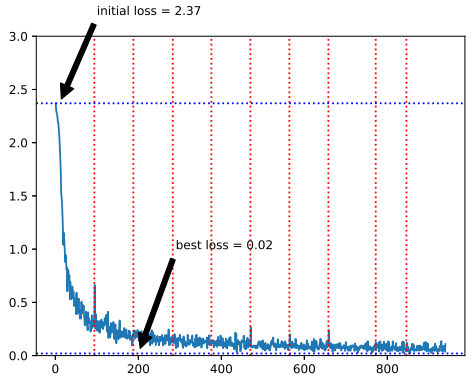

Fig. 4. Training loss for gradient-based convolutional network training. Dotted red lines indicate transitions between epochs.

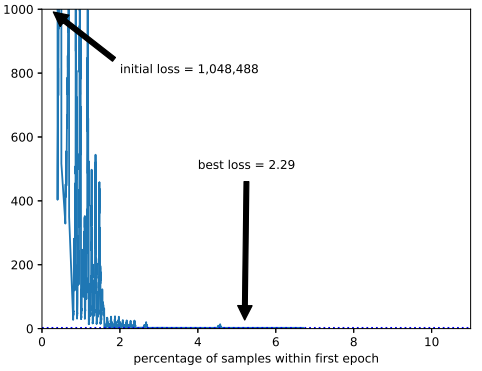

Fig. 5. Training loss for the first experimental case. Loss values dropped significantly after the first few batches, and then plateaud at around 2.3. It is noteworthy that the SDN-based implementation was able to find decent weight configurations without any explicit initialization, even after the first few batches.

[18]. PyTorch is an open-source Python library that has been gaining traction in the past few years, due to its effective implementation of GPU-based tensor operations and its flexible automatic differentiation model that allows for the network architecture to change dynamically, even while training.

A. Network architecture

The example network used for the experiments was a convolutional neural network with 4 layers:

- Layer 1: A 2-dimensional convolutional layer with a kernel size of 5, and 5 output layers (yielding a total of 250 weights)
- Layer 2: A 2-dimensional convolutional layer with a kernel size of 5, and 20 output layers (yielding a total of 5,000 weights)
- Layer 2b: A dropout layer with no modifiable weights
- Layer 3: A linear layer with a rectified linear output, consisting of 16,000 weights
- Layer 4: A linear layer with a softmax output, consisting of 500 weights

The network also contained bias values, but in the SDN case, only the weights were trained. Hence, the number of trainable parameters was in the order of the tens of thousands.

B. Comparison of results

The evolution of training errors for the normal, gradient-based training procedure can be seen on Figure 4. A comparison was performed between the gradient-based training results and two experimental cases as follows.

1) Results using SDN with partition-based hierarchy: In the first experimental case, the weights in the convolutional neural network were broken into partitions based on the layer in which they were defined. As a result, 4 different SDN models were used.

The hyperparameters of the SDN models were chosen such that layers with more weights were characterized by longer cycles and a higher refresh rate. Specifically, the SDNs corresponding to the 4 layers had cycle lengths of 4, 80, 160 and 8, respectively (with the cycle length of 160 attributed to layer 3, which contained 16,000 weights). Similarly, the refresh periods were 16, 4, 2 and 8, respectively (meaning that the SDN corresponding to layer 3, for example, was updated after every 2 batches, while the SDN corresponding to layer 1 was updated once every 16 batches).

For each of the SDN models, the *step_size* was set to 0.05, while the vectors $\mathbf{p}^{(0)}$ and $\mathbf{p}^{max}$ were initialized with random values from -0.1 to 0.1, and with values of either -0.1 or 0.1, respectively.

The logic of training was as follows: upon each new batch of 64 training samples, the SDNs whose refresh period ended were refreshed, and a new cycle was generated. Then, a weight configuration was generated for each activation of the SDN with the longest cycle was generated (the activations of those SDNs which generated shorter cycles were upsampled to match the number of output configurations). After every 10 batches, the best loss value was recorded.

A plot of the first results is shown on Figure 5. It is noteworthy that although the partition-based search initially had no conception of what constituted a good initialization of weights, as well as no conception of even the scale at which good weights may arise, it was able to find configurations after just a few batches which were on par with results obtained in the first batches of gradient-based training.

2) Results using SDN with partition-based hierarchy and a hierarchical SDN for step size parameters: One idea original presented in [7] is to create hierarchical SDNs in which top-level SNDs are responsible for iterating through different meta-parameter values. In the second experiment, such an SDN model was used to determine

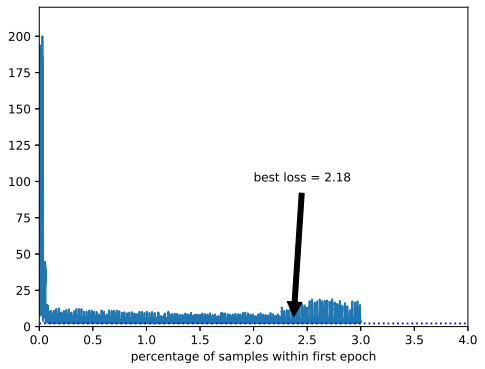

Fig. 6. Training loss for the second experimental case. Loss values dropped even earlier than in the first experiment, and then plateaud at around 2.2.

the step size for each of the SDNs representing the 4 layers.

A plot of the first results is shown on Figure 6. In this case, the loss value dropped even more quickly, but still plateaud around the minimal value. As future work, it would be worth investigating some ways in which SDNs networks may alter their hyperparameters more effectively.

V. Conclusions

Evolutionary and neural network based parametric optimization approaches are two separate worlds, each with their own set of assumptions and limitations. Importantly, there exist problem domains where neither approach is in itself effective. This can be due to the fact that there may be no natural way to encode candidate solutions as evolutionary genotypes with respect to which the operators of mutation and recombination make practical sense. In other cases, it may be difficult to find a globally defined loss function that is consistent, can be effectively computed and is also differentiable. In such cases, the Spiral Discovery Network model and its extensions proposed in this paper can be used as an alternative approach that can be used with coarse-grained feedback evaluations and without any explicit gradient information. Preliminary experiments on training a traditional convolutional neural network architecture show promising results. The convergence properties of SDNs would be worth investigating further in such contexts.

References

[1] P. Baranyi, A. Csapo, and G. Sallai. Cognitive Infocommunications (CogInfoCom). Springer International Publishing, 2015.

[2] P. Baranyi and A. Gilanyi. Mathability: emulating and enhancing human mathematical capabilities. In Cognitive Infocommunications (CogInfoCom), 2013 IEEE 4th International Conference on, pages 555–558. IEEE, 2013.

[3] P. Biró and booktitle=Cognitive Infocommunications (CogInfoCom), 2015 6th IEEE International Conference on pages=111–114 year=2015 organization=IEEE Csernoch, M. The mathability of computer problem solving approaches.

[4] K. Chmielewska, A. Gilányi, and A. Łukasiewicz. Mathability and mathematical cognition. In Cognitive Infocommunications (CogInfoCom), 2016 7th IEEE International Conference on, pages 000245–000250. IEEE, 2016.

[5] A. Csapo and P. Baranyi. The Spiral Discovery Method: an Interpretable Tuning Model for CogInfoCom Channels. Journal of Advanced Computational Intelligence and Intelligent Informatics, 16(2):358–367, 2012.

[6] A.B. Csapo. The spiral discovery network as an automated general-purpose optimization tool. Complexity, 2018:1–8, 2018.

[7] A.B. Csapo. Hierarchical spiral discovery networks for multi-layered exploration-exploitation tradeoffs. Acta Polytechnica Hungarica, In press.

[8] A. Dobrovsky, U.M. Borghoff, and M. Hofmann. An approach to interactive deep reinforcement learning for serious games. In Cognitive Infocommunications (CogInfoCom), 2016 7th IEEE International Conference on, pages 000085–000090. IEEE, 2016.

[9] P. Domingos. The master algorithm: How the quest for the ultimate learning machine will remake our world. Basic Books, 2015.

[10] P. Földiák. Learning invariance from transformation sequences. Neural Computation, 3(2):194–200, 1991.

[11] I. Horvath and A. Sudar. Factors contributing to the enhanced performance of the maxwhere 3d vr platform in the distribution of digital information. Acta Polytechnica Hungarica, 15(3):149–173, 2018.

[12] J. Katona and A. Kovari. Examining the learning efficiency by a brain-computer interface system. Acta Polytechnica Hungarica, 15(3), 2018.

[13] B. Lampert, A. Pongracz, J. Sipos, A. Vehrer, and I. Horvath. Maxwhere vr-learning improves effectiveness over clasiccal tools of e-learning. Acta Polytechnica Hungarica, 15(3), 2018.

[14] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010.

[15] M. Minsky. Steps toward artificial intelligence. Proceedings of the IRE, 49(1):8–30, 1961.

[16] A. Minzoni, E. Mounoud, and V.A. Niskanen. A case study on time-interval fuzzy cognitive maps in a complex organization. In Cognitive Infocommunications (CogInfoCom), 2017 8th IEEE International Conference on, pages 000027–000032. IEEE, 2017.

[17] V.A. Niskanen. Application of fuzzy cognitive maps to business planning models. In Theoretical Advances and Applications of Fuzzy Logic and Soft Computing, pages 119–127. Springer, 2007.

[18] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In NIPS-W, 2017.

[19] J. Schmidhuber. Deep learning in neural networks: An overview. Neural networks, 61:85–117, 2015.

[20] D. Sik, K. Csorba, and P. Ekler. Implementation of a geographic information system with big data environment on common data model. In Cognitive Infocommunications (CogInfoCom), 2017 8th IEEE International Conference on, pages 000181–000184. IEEE, 2017.

[21] J. Stone and A. Bray. A learning rule for extracting spatio-temporal invariances. Network: Computation in Neural Systems, 6(3):429–436, 1995.

[22] H. Takagi. Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. Proceedings of the IEEE, 89(9):1275–1296, 2001.

[23] H. Takagi and H. Iba. Interactive evolutionary computation. New Generation Computing, 23(2):113–114, 2005.

[24] Á. Török, Z.G. Török, and B. Tölgyesi. Cluttered centres: Interaction between eccentricity and clutter in attracting visual attention of readers of a 16th century map. In Cognitive Infocommunications (CogInfoCom), 2017 8th IEEE International Conference on, pages 000433–000438. IEEE, 2017.

[25] L. Wiskott and T.J. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. Neural computation, 14(4):715–770, 2002.