Real World Examples of Agent based Decision Support Systems for Deep Learning based on Complex Feed Forward Neural Networks

Harald R. Kisch and Claudia L. R. Motta
NCE, Universidade Federal do Rio de Janeiro, Av. Pedro Calmon, 550 - Cidade Universitria, Rio de Janeiro, Brazil

Keywords: Deep Learning, Neural Network, Feed Forward, Muilti-agent, Complex Systems, Decision Support.

Abstract: Nature frequently shows us phenomena that in many cases are not fully understood. To research these phenomena we use approaches in computer simulations. This article presents a model based approach for the simulation of human brain functions in order to create recurrent machine learning map fractals that enable the investigation of any problem trained beforehand. On top of a neural network for which each neuron is illustrated with biological capabilities like collection, association, operation, definition and transformation, a thinking model for imagination and reasoning is exemplified in this research. This research illustrates the technical complexity of our dual thinking process in a mathematical and computational way and describes two examples, where an adaptive and self-regulating learning process was applied to real world examples. In conclusion, this research exemplifies how a previously researched conceptual model (SLA process) can be used for making progress to simulate the complex systematics of human thinking processes and gives an overview of the next major steps for making progress on how artificial intelligence can be used to simulate natural learning.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

Many of our technical innovations are inspired by nature. Most of the things we categorize in biological evolution, like how human brains works, are still difficult to achieve with our current technical skills. Backfiring introduced by Hebb, 1949 with "Firing together, wired together" and nowadays called backpropagation or Forward- or Reverse-mode differentiation (Ochs et al., 2016), which means reducing many neural network paths by finding and propagating intermediate results that significantly reduce path-reassemblings or different or additional path-assimiliations (Krizhevsky et al., 2012). Recently, the Forward-mode differentiation (tracking how one input affects every node) and the Reverse-mode differentiation (tracking how every node affects one output) to mathematically evaluate derivatives were presented (Ochs et al., 2016). As an advantage of implementing Reverse-mode differentiation it is possible to get all the derivatives that produce the output network streams with one single computation. This reduces the computational costs of so called first-order optimization algorithms within learning functions e.g. by finding a local minimum or maximum. The other way around, implementing Forward-mode differentiation means getting all derivatives that produce the input with one single computation. Thus, we can choose the computation depending on which operation performs better with respect to the input or output sequences. If there is a computation with a lot of outputs, the chain rule is to use Forward-mode differentiation to create deep feedforeward neural networks. If there is a computation with a lot of input, the chain rule is to use Reverse-mode differentiation to create neural networks, whose derivatives are flowing backwards through the model. The replacement of the logistic or the hidden networks is represented through an analog value, which has its own input and output gates that control when inputs are allowed to influence the output. These gates have their own learned weights on connections and memories that are stored in the previous step. In the past, it seemed unlikely that an exact algorithm can be developed to perform probabilistic inference efficiently over all classes of belief networks (Cooper, 1990). However, 20 years later the theory of discrete calculus was introduced (Grady and Polimeni, 2010), showing that it can be applied in many fields and disciplines by unifying approaches for data analysis and content extraction. These extractions can speed up the optimization of neural networks by the amount of input or output se-

Kisch, H. and Motta, C.
Real World Examples of Agent based Decision Support Systems for Deep Learning based on Complex Feed Forward Neural Networks.
DOI: 10.5220/0006307000940101
In *Proceedings of the 2nd International Conference on Complexity, Future Information Systems and Risk (COMPLEXIS 2017)*, pages 94-101
ISBN: 978-989-758-244-8
Copyright © 2017 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

quences in recurrent neural network (RNN) streams. This leads to a faster learning of long term dependencies in Long Short Term Memory (LSTM) Recurrent Neural Networks (Le et al., 2015). Hinton et. al (2015) also exemplifies the solution of some challenging problems like sequence regression, presentation and performance increase on hidden layer establishment at initialization time. Some examples of this approach can also be found in handwriting recognition (Graves et al., 2009), handwriting generation (Graves, 2013), speech recognition (Graves et al., 2013), (Graves and Jaitly, 2014), machine translation (Luong et al., 2014), (Bahdanau et al., 2014) and sequence to sequence mapping (Sutskever et al., 2014). In this article, we use the theory of Dual Thinking (Scheffer et al., 2015) as a meta model and combine it with a previous work on a human thinking model based on a neural network called the 6-Step Learning Adaption process (SLA) (Kisch and Motta, 2015). The main idea of SLA is the step-wise, agent oriented approach to collect, associate, operate, define, store and transfom knowledge map fractals in a tool-validated conceptual model.

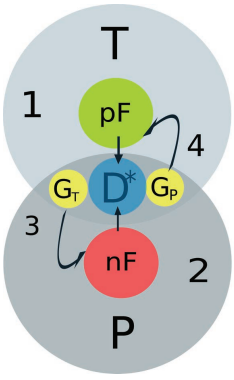

Figure 1: Dual Thinking.

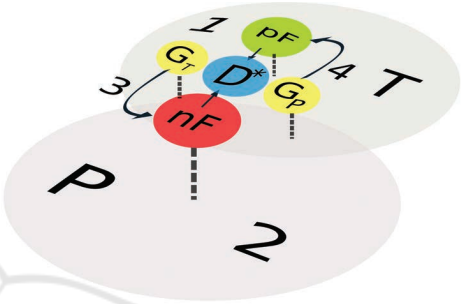

Figure 2: Dual Thinking Layer.

2 METHODOLOGY

Many scientists consistently conclude that intuition and reasoning is an essential part of our thinking process (Scheffer et al., 2015). In relation to the SLA process intuition can be realized by mapping gate-associations, where gates are definitions of neuronal sub-networks according to Sporns motifs (Sporns and Koetter, 2004). On the other hand reasoning in the SLA-Process are the operations on the definitions of neural sub-networks which lead to definition-increments into the overall learning map. Each map fractal can span out the whole network from which it was defined. Each node in the network saves the state of any other node related to the definition. This means the collection at the beginning, the associations, the operations that were made and also the transformation including the previously defined results and predictions for the changes of the next transformation based on the positive and negative feedback out of the probability testing network in a whole.

Notes: P = Product (Reasoning), T = Testing (Intuition), nF = negative Feedback, pF = positive Feedback, G_T = Testing-Gate, G_P = Product Gate, D^* = Current Definition

With both processes for reasoning and testing combined, there are some feedback-networks with positive and negative feedback on the actual thinking process called negative Feedback (nF) and positive Feedback (pF) as shown in Figure 1 in the center of the top and of the bottom circle, respectively. Negative feedback (nF) happens primarily in the Intuition-Test Neuronal Network (T) and pF happens primarily in the Reasoning-Neural Network (P) as shown in Figure 1 on the top circle and on the bottom circle. Feedback is the definition of a previous match of associations and can be positive (many matches) or negative (no matches). The feedback of many neural networks in this approach is generally stored as result and used to reach group consensus according to Liao and Ji (Liao and Ji, 2014).

3 RESULT

Many approaches, which are used to mathematically describe group consensus (Ji et al., 2014) in neural networks need many restrictions because they cannot be applied directly to real world szenarios. This is why the formalization approach in this article is a general overview to explain the conceptual layer model of figure 1. In the following, the formalization of the Dual Thinking process illustrated in Figure 1 and 2 is conceptional formalized in general and meant to be used in big data computation scenarios.

Implementation Setup Formalization.

$$ATT := JSON \in \{Array, Function, Object\} \quad (1)$$

$$Spec := Object \in \{key : value\} \quad (2)$$

$$DOC := \bigcup_{n=1}^{\infty} ATT + Spec + Rev\# \quad (3)$$

$$DB := \bigcup_{n=1}^{\infty} DOC \quad (4)$$

$$N := \bigcup_{n=1}^{\infty} DB \quad (5)$$

Note: ATT = Attachment, $Spec$ = Specification, DOC = Document, DB = Database, N = Node, $Rev\#$ = unique revision number, which is used to build a concurrent versioning of any knowledge map fractal. This is mandatory for the transformation step in the SLA process to extract resonances of positive and negative feedback (explained later on) without having to change the knowledge at all. On negative feedback for example, the state of the best feedback resonance will be recovered using the saved $Rev\#$.

Model Formalization.

$$NN := \bigcup_{n=1}^{\infty} N \quad (6)$$

$$M := \bigcup_{n=1}^{\infty} NN \quad (7)$$

$$G := \bigcup_{n=1}^{\infty} M \quad (8)$$

$$D := \bigcup_{n=1}^{\infty} G \quad (9)$$

$$S := \bigcup_{n=1}^{\infty} D \quad (10)$$

Note: NN = Neural Network, M = Motif, G = Gate, D = Definition, S = Stack item
The stack S is saved in every node N to span out the whole NN to further recure the inputs leading to the definition D. We can say that a reverse function which returns all the inputs I of a neural network is defined as following:

$$I = f(S_D^G) \rightarrow NN_G \cup D_G \quad (11)$$

The other way around, the reverse function D_G^* is the weighted conjunction between G_T^* and G_P^* producing the Output O and can defined as:

$$O = f(G_T \cup G_P) \rightarrow D^* \quad (12)$$

Any NN consist of a testing network T which represents the imagination part of our thoughts and a producing network P which represents the reasoning part of our thoughts. The conclusions (definitions D) of T and P together are saved in hidden feedback networks nF and pF .

$$nF = f(NN_T) \rightarrow O(G_T); NN_T, G_T \in D^* \quad (13)$$

$$pF = f(NN_P) \rightarrow O(G_P); NN_P, G_P \in D^* \quad (14)$$

Figure 1 illustrates (11) to (14). In this illustration, each circle is meant to be a neural network or a part of a neural network Motif (Sporns and Koetter, 2004) itself. The Sequences $S1$ to $S4$ executes the Sequence functions s_1 and s_2 as following:

$$S1 := f(T) \in s_1 \quad (15)$$

$$S2 := f(P) \in s_1 \quad (16)$$

$$S3 := f(G_T) \in s_2 \quad (17)$$

$$S4 := f(G_P) \in s_2 \quad (18)$$

Sequences $S1$ to $S4$ in figure 1 are followed by sequences s_1 and s_2, which can be defined as following:

$$s_1 := f(T) \cup f(P); T, P \in D^* \quad (19)$$

$$s_2 := f(G_T) \cup f(G_P); G_T, G_P \in D^* \quad (20)$$

The feedback functions nF and pF are defined as following where nF is meant to be the negative part and pF the positive part of the feedback:

$$nF := f(T) \cup f(G_T); T, G_T \in D^* \quad (21)$$

$$pF := f(P) \cup f(G_P); P, G_P \in D^* \quad (22)$$

The feedback functions conclude, that the following definition D^n is a neural network NN of positive inputs I_P out of the current network M^* transformed by the test network T, producing products P of positive feedback gates $nF(G_P)$ in disjunction with all negative feedback gates $nF(G_T)$, so that the resulting function for the next definition can be used as following:

$$D^n := M^* \prod_{I_P=1}^{n} T\Big(pF(G_P) \cap nF(G_T)\Big); I_P \in T(f(S_D^G)) \quad (23)$$

In most cases a complex problem needs an appropriate environment, in which the problem can be solved. This environment needs training on the specified inputs as well as a finishing condition with defined dropouts to prevent overfitting (Srivastava et al., 2014).

4 DIFFERENCE TO TYPICAL NEURAL NETWORK DESIGN

Research in the area of neural brain functions already started centuries ago i.e. with the probabalistic model of a perceptron (Rosenblatt, 1958). Rosenblatt created the fundament on which Feed-Foreward neural Networks (FFNN) are based. In general, they are based on learned datasets of what is wanted to have coming out of the network. This is also called supervised learning (Jordan and Rumelhart, 1992).

On the other hand, unsupervised learning is commonly understood as putting inputs into the network and let the network fill out the missing parts based on what was previously trained (Hofmann, 2001). Boltzmann machines (Hinton and Sejnowski, 1986), Markov chains (Kani and Ardehali, 2011) and Hopfield networks (Hopfield, 1982) are more similar to the SLA model because each node is related to all other nodes and also some basic responsibilities are defined such as input, output and restrictions to some nodes but not all of them like in the SLA model. Any of the mentioned conceptual models and also other more modern Encoders (Gregor et al., 2015) are all based on the inputs and the expected outcome. This is the main difference to the SLA process. The SLA model processes each constellation of the network without an expected outcome. It transforms the network into domain clusters to get outcomes related to specific universals which are taxonomy-based significant to the provided inputs and of course take any changing aspect into account by the transformation step and relate it to all of the other aspects in a probabalistic way during the association step of the SLA processing chain.

5 APPLICATION TO REAL WORLD EXAMPLES

The SLA process was tested in two different environments. First, it was tested if it can produce relevant outputs in form of recommendations in the field of technical mechanics and integrated circuit parts of the public transport industry. The second environment, football game predictions, was selected to be a completely different context compared to the first simulation. The inputs and training data as well as the output of the second simulation was selected to be completely different. Football game predictions were considered as being easier to adapt and to test and are easier to understand than technical concepts in the mechanical and electronical industry.

5.1 Automatically Finding Obsolescence Management Solutions

According to Bartels (Bartels, 2012), integrated circuits and electronical parts are obsolete if the industry is not able to have them delivered anymore or they are not needed anymore e.g. because of law restrictions or technological evolution. Many railway companies have the same problems and need to find a way to replace obsolete parts to keep public transport operable longer then estimated without obsolescence management. At this point, obsolescence management focuses on the lifetime extension of any kind of technical construction.

Simulation design I.
For the communication between railway companies on the subject of obsolescence management, an internet platform has been developed. Currently about 50 people from 20 different railway companies communicate on different obsolescence management tasks through the platform every day. Figure 3 shows how the obsolescence cases have grown over time. Important to know is that the cases come from different modes of transportation e.g. trains, metro, tram, bus. Focusing all modes of transportation simultaneously is useful because parts and internal components can be identical across different modes of transportation, e.g. train, bus, metro, etc. However, the aim of the SLA process in this area is to predict which parts can be replaced with a commonly accepted solution automatically at the time a new obsolescence case was entered. Entered cases are saved according to the international system of railway labeling EN 15380-2, thus the matching algorithm can learn based on international standard input sequences provided by user inputs. The procedure of doing obsolescence management is also conform with the international standard DIN EN 62402. This enables the algorithms to analyze user behavior and learn which recommendations are most relevant to the user at a specific time point.

Methods applied I.
To accomplish the main tasks in this area the learning algorithm learns from user inputs and produced

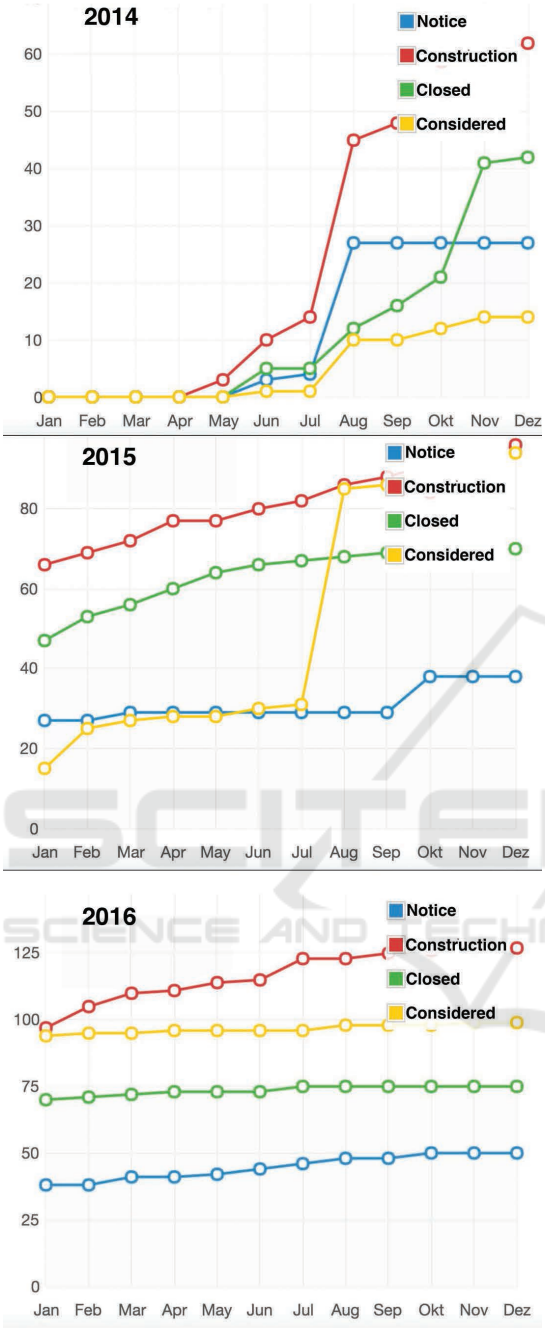

Figure 3: Obsolescence Management Data Growth.

data entered by the user or generated through the SLA process for a specific environment or product family. If the user enters a complete obsolescence case with its solution included, the SLA process searches for similarities between different other obsolescence cases and matched versions, part numbers and other criteria applicable for the identification, location and the specific environment of a given part at collection time. The association step binds all collected items together into environments or product families in order to operate on them in the next step. At operation time each of the associations were mapped to different specific patterns previously stored and related to the specified product families and component interfaces. If nothing was found, the collections and associations are transformed using positive and negative feedback loop algorithms combined with possibly missing data to force a pattern match. Otherwise if something significant were found, the grade of relevance for the solution gets stored for further usage in next iterations.

Results I.
The simulation has shown that in the field of obsolescence management the SLA process was less useful in the specified area of component part mappings. We figured out that we can make good suggestions to the user which components could fit together, unfortunately there was not enough data to work on relevant pattern matching. We have not been able to construct as much relevant data across the different railway environment and product families to get significant mapping patterns which are pointing on solutions the industry can use in practice. The most difficult area of transforming and operating data was the incompatible design of integrated components. At construction time constructors did not consider the compatibility across the interfaces of other, mostly competitive integrated circuits or mechanical components of other railway or public transportation vehicle part creators. This kind of investigation seems to have much better results if creators are forced to design compatible interfaces by governance, law or international standards.

5.2 Football Game Prediction

Another approach for SLA process application is the area of football game statistics to predict a football teams performance compared to other football teams during an official football match across nearly 60 different leagues around the world. This simulation was selected out of many different options because of the grade of difference to the study previously described, to test the applicability of the SLA process for different fields of data computation and different kind of predictions or expected outcomes.

Simulation Design II.
The statistical baseline for game predictions of the last three football seasons in the learning part of the SLA process was retrieved from a public available database via http://www.sofascore.com (Sofa). The Application Programming Interface (API) of Sofa

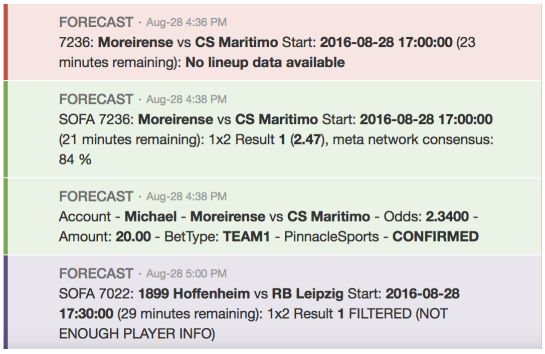

Figure 4: Bet Filtering.

provides all relevant data e.g. on matches, scores, teams and players of several football leagues. The SLA Model was designed to work in the same way as a professional football expert would bet. Building up domains for leagues, teams, and regional history, then collecting data about past game results of the competing teams (Step 1 Collection), players, judges and observing the motivation regarding the current league rank of relevant teams. Operations (Step 3 Operation) have been applied to calculate a player score for each possible team formation during the match (Step 2 Association). Each constellation was saved (Step 5 Storage) as a relevant team score (Step 4 Definition) and was initiated as an own transformable network (Step 6 Transformation). Any relevant information was then calculated for each of the possible team formations to calculate a player score for each player in different field positions and their relevant opponents in the attack-, middle-, and defense-field. The results have been saved in a meta network in which all predictions for one football match have been placed. Any constellation of the variables did predict a specific chance of competing the opponent in a simple value based on randomly generated data for the specific match situations the players usually have.

Methods applied II.
To predict which team will win in a specific football match, different methods have been applied. The prediction which team will win was interpreted out of a percentage rate of consensus for each of the meta network values and generated data on which they are based on. The illustration in Figure 4 shows accepted bets and bets which have been filtered out because they did not match the createria to have more than 80% meta network consensus in a rising amount of different team constellations and in-match player situations. Predictions cannot be placed if the relevant data is not available. Usually the player formation data is available 30 minutes before the game starts. An automated algorithm extracts the data from Sofa

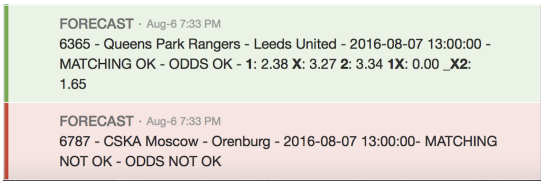

Figure 5: Bet Quote Results.

and hands it over it to the SLA System. Within seconds it is calculated if the bet is filtered out or will enter the betting queue. There are three options for placing a bet as illustrated in Figure 5. The SLA System was learned to find out the best betting option on a previously calculated range based on the Gaussian bell curve concerning the bet quotes, player and team constitution as well as different approaches to reduce wrong predictions because of outlying values. A bet was always only placed if all meta networks gave a consensus of at least 80%, otherwise the bet for the match was rejected. Initial simulations indicated unreliable predictions, extrem outliers or many missing data which is why for several leagues it was decided not to bet at all.

Results II.
Figure 7 shows how the simulation did predict a concrete income for a comparable low amount of possible bets at a computational low risk. The discrepancy between theory and practice regarding football game predictions, based on the SLA algorithm learned on real seasonal data from the past, becomes clear when comparing to the bet reality illustrated in Figure 6, where earnings decrease with increasing number of bets. Thus, football game predictions are harder to predict if they come from complex automated algorithms. The analysis of the outcomes has shown that there were huge amounts of football matches where, according to the statistics, stronger teams lost the match. Compared to the Obsolescence Management study, the main differences are the reliability and the amount of patterns found. In football game predictions there where significantly less patterns found and these where also less reliable compared to the area of recommendation engines in the field of public transportation technology.

6 DISCUSSION

Analytical Data Processing for recommender systems (Manouselis et al., 2011) like those systems developed for e-commerce decision making (Kim and Srivastava, 2007) or machine translation (Bahdanau et al., 2014) or acustical noice reduction (Srivastava

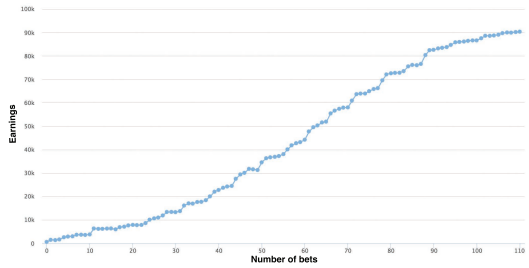

Figure 6: Bet Simulation.

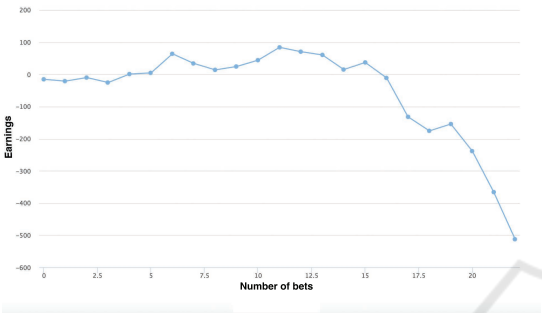

Figure 7: Bet Reality.

et al., 2014) already use machine learning algorithms based on neural networks. They are all extremely specialized and cannot be used in other environments as they have been made for. Ubiquitous computing or at least the Social Internet of Things (SIoT) (Atzori et al., 2012) will hold many autonomous complex systems with its own business logics and their own intelligence in a peer-to-peer environment of similar autonomous systems sharing information all together in different environments. Environments change very fast or need to cover unpredictable changes. Any machine learning process should be able to learn from the inputs and outcomes and thus to reconfigure themselves to be able to react on unexpected environmental changes. In comparison to other research, the SLA process, based on biological neuron functions is similar to the approaches of combining artificial intelligence with widely approved biological and chemical processes like DNA Sequencing (Ezziane, 2006) or the classification of protein structures (Suhrer et al., 2009) as both approaches extract knowledge from an enormous amount of data. Further investigations are necessary to also cover following aspects:

First, the ”State of Mind” concept, how different Neuronal Networks can be used to represent something similar to the ”state of the mind” for a specific recognition or the polarity between sentiments (Poria et al., 2014). Second, ”Priming” concept, how the element association can be marked as complete, abstract, stable or has a significant convergence to a specific state defined previously of being important for the current reasoning process based on the consensus of positive and negative feedback loops.
Third, ”Balancing” concept between positive and negative feedback ($nF <> pF$) to store a specific definition (D) for later use. Fourth, theoretical complexity and performance through efficient cache, transience and garbage collection models are not focused yet. Last, ”Avoiding” concept in order to shrink, expand or test the neural networks in order to rise effectiveness of the learning function (e.g. reducing errors faster) (Srivastava et al., 2014).

7 CONCLUSION

The combination of both, the SLA and the Dual Thinking process was considered to be a good fit for making progress in simulating human thinking processes for technology enhanced learning in theory. But both of the studies exemplified in this paper show that something of higher significance is missing to compare computational simulation processes, like SLA with natural learning. Machine learning algorithms, especially in the manner of imagination and reasoning, will improve many modern life scenarios in the near future. Decision making based on the behavioral history of complex environments like in the SLA process could take a significant part of it. These decision making processes definitely need further investigation and deeper statistical analysis for positive and negative feedback relevances. The level of significance need to be saved in the knowledge map as well. Only in this way it is possible to know the share of a knowledge map fractal on the decision making process. This enables change prediction in the transformation processing step and could lead to more efficient conclusions. It seems there exists a necessity for statistical baselines and standardized taxonomy in each knowledge transformation step. Operations, selections, associations and transformations should be based on statistical data analysis and need more data as used in both approaches exemplified in this research.

REFERENCES

Atzori, L., Iera, A., Morabito, G., and Nitti, M. (2012). The social internet of things (siot)–when social networks meet the internet of things: Concept, architecture and network characterization. *Computer Networks*, 56(16):3594–3608.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bartels, B. (2012). *Strategies to the prediction, mitigation and management of product obsolescence*. Wiley.

Cooper, G. F. (1990). The computational complexity of probabilistic inference using bayesian belief networks. *Artificial intelligence*, 42(2):393–405.

Ezziane, Z. (2006). Applications of artificial intelligence in bioinformatics: A review. *Expert Systems with Applications*, 30(1):2–10.

Grady, L. J. and Polimeni, J. (2010). *Discrete calculus: Applied analysis on graphs for computational science*. Springer Science & Business Media.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Graves, A. and Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1764–1772.

Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., and Schmidhuber, J. (2009). A novel connectionist system for unconstrained handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(5):855–868.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE.

Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., and Wierstra, D. (2015). Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*.

Hinton, G. E. and Sejnowski, T. J. (1986). Learning and relearning in boltzmann machines. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1:282–317.

Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine learning*, 42(1-2):177–196.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.

Ji, L., Liu, Q., and Liao, X. (2014). On reaching group consensus for linearly coupled multi-agent networks. *Information Sciences*, 287:1–12.

Jordan, M. I. and Rumelhart, D. E. (1992). Forward models: Supervised learning with a distal teacher. *Cognitive science*, 16(3):307–354.

Kani, S. P. and Ardehali, M. (2011). Very short-term wind speed prediction: a new artificial neural network–markov chain model. *Energy Conversion and Management*, 52(1):738–745.

Kim, Y. and Srivastava, J. (2007). Impact of social influence in e-commerce decision making. In *Proceedings of the ninth international conference on Electronic commerce*, pages 293–302. ACM.

Kisch, H. R. and Motta, C. L. R. (2015). Model of a neuron network in human brains for learning assistance in e-learning environments. In *Proceedings of the 7th International Conference on Computer Supported Education*, pages 407–415.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Le, Q. V., Jaitly, N., and Hinton, G. E. (2015). A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

Liao, X. and Ji, L. (2014). On pinning group consensus for dynamical multi-agent networks with general connected topology. *Neurocomputing*, 135:262–267.

Luong, M.-T., Sutskever, I., Le, Q. V., Vinyals, O., and Zaremba, W. (2014). Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*.

Manouselis, N., Drachsler, H., Vuorikari, R., Hummel, H., and Koper, R. (2011). Recommender systems in technology enhanced learning. In *Recommender systems handbook*, pages 387–415. Springer.

Ochs, P., Ranftl, R., Brox, T., and Pock, T. (2016). Techniques for gradient-based bilevel optimization with non-smooth lower level problems. *Journal of Mathematical Imaging and Vision*, pages 1–20.

Poria, S., Cambria, E., Winterstein, G., and Huang, G.-B. (2014). Sentic patterns: Dependency-based rules for concept-level sentiment analysis. *Knowledge-Based Systems*, 69:45–63.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Scheffer, M., Bascompte, J., Bjordam, T. K., Carpenter, S. R., Clarke, L., Folke, C., Marquet, P., Mazzeo, N., Meerhoff, M., Sala, O., et al. (2015). Dual thinking for scientists. *Ecology and Society*, 20(2).

Sporns, O. and Koetter, R. (2004). Motifs in brain networks. *PLoS Biology*, 2(2):e411.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Suhrer, S. J., Wiederstein, M., Gruber, M., and Sippl, M. J. (2009). Cops a novel workbench for explorations in fold space. *Nucleic acids research*, 37(suppl 2):W539–W544.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.