A Pattern Recognition Model of Python Programming using Artificial Neural Network via NeMo

M.Janardhan, M.Srilakshmi, S Prem Kumar

Abstract: Background/Objectives: In the field of software development, the diversity of programming languages increases dramatically with the increase in their complexity. This leads both programmers and researchers to develop and investigate automated tools to distinguish these programming languages. Different efforts were conducted to achieve this task using keywords of source codes of these programming languages. Therefore, instead of using keywords classification for recognition, this work is conducted to investigate the ability to detect the pattern of a programming language characteristic by using NeMo(High-performance spiking neural network simulator) of neural network and testing the ability of this toolkit to provide detailed analyzable results. Methods/Statistical analysis: the method of achieving these objectives is by using a back propagation neural network via NeMo based on pattern recognition methodology. Findings: The results show that the NeMo neural network of pattern recognition can identify and recognize the pattern of python programming language with high accuracy. It also shows the ability of the NeMo toolkit to represent the analyzable results through a percentage of certainty. Improvements/Applications: it can be noticed from the results the ability of NeMo simulator to provide beneficial platform for studying and analyzing the complexity of the backpropagation neural network model.

Keywords: NeMo, Pattern recognition, artificial neural network, Backpropagation neural network.

I. INTRODUCTION

NeMo (Neural Modules) is a Python framework-agnostic toolkit for creating AI applications through re-usability, abstraction, and composition. NeMo is built around neural modules, conceptual blocks of neural networks that takes typed inputs and produce typed outputs. Such modules typically represent data layers, encoders, decoders, language models, loss functions, or methods of combining activations. NeMo makes it easy to combine and re-use these building blocks while providing a level of semantic correctness checking via its neural type system.

Manuscript published on January 30, 2020.
* Correspondence Author
M.Janardhan*, Associate Professor, Department of Computer Science and Engineering, G.Pullaiah College of Engineering and Technology (Autonomous).
M.Srilakshmi, Assistant Professor, Department of Computer Science and Engineering, G.Pullaiah College of Engineering and Technology (Autonomous).
Dr.S Prem Kumar, Professor & Dean, Department of Computer Science and Engineering, G.Pullaiah College of Engineering and Technology (Autonomous).

In the last decade, a wide range of programming languages for a variety of tasks have been created in the software development field)philip Mayer ‹April 2015(. This diversity makes it difficult for new students and developers to recognize the exact programming language that been used in complex systems.

Especially in the systems that require using a combination of programming languages such as python, Java or Ruby)philip Mayer ‹April 2015(. Therefore, it would be beneficial to develop a tool for identifying programming language codes based on its pattern. One of the attempts to achieve this task is conducted by M. Robson)Montenegro ‹ (2016 and (Jyotiska Nath Khasnabish, 2014) through training a neural network model to classify programming codes based on its language. According to M. Robson)Montenegro(2016 ‹ , this classifier identifies programming languages based on syntax codes in the form of words. Robson suggests using the characters' patterns of the programming language instead of these keywords. Therefore, instead of using the classification of keywords in the codes for different programming languages, this work aims to investigate the ability of back propagation neural network (BNN) to identify and recognize the programming language (python) based on the pattern of each particular code characteristics. This paper also aims to investigate the ability of NeMo, a neural network simulation, to represent analyzable results.

II. BACKGROUND

2.1 Pattern Recognition

Pattern recognition can be defined as a methodology of designing systems that can identify or classify patterns in complex environment)Sargur N. Srihari ‹ (1993. It also “can be seen as a classification process”)SALIBA(2014 ‹. It aims to study and monitor environment for a potential pattern and make a proper decision about it)Jayanta Kumar Basu(2010 ‹. According to Sharma and Kaur)Priyanka Sharma(2013 ‹ the basic algorithm of pattern recognition can be illustrated in Figure 1

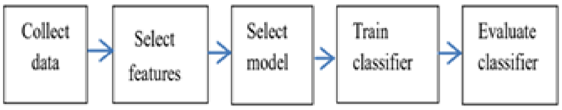

Figure 1. Algorithm of pattern recognition)Priyanka Sharma(2013 ،**.**

As shown in Figure 1, the first part of this algorithm is collecting data then selecting the features from the data to be recognized. After preparing input data, suitable recognition model is selected. This model is trained to recognize the potential pattern. Finally, the system is evaluated to check it behavior. In addition, Based on Sharma and kaur classification, the main model of pattern recognition are statistical model, syntactic model, template matching model, and Artificial neural network which is characterized for “ the ability to learn complex nonlinear input- output”)Priyanka Sharma(2013 ،.

2.2 Artificial Neural Network

Essentially, the idea of artificial neural network (ANN) is based on the concept of how the information processes inside humans and animals brains. This concept can be oversimplified as a complex network of trillions of nerve cells interconnected with each other via pulses called action potentials)Smith(1997 ، . According to Steven W. S.)Smith(1997 ، , ANN aims to mimic this process as much as possible. Which means mimicking the most important ability in human mind, which is the ability of learning. This is differs from the linear algorithm of regular machine methodology to solve problems. In other words, ANN can be simply defined as computer algorithms that consist of simple entities interconnect with each other to form an interaction of behaviour in response to different states of input)Gurney(1997 ،. It is structured in the form of layers each of which is consisting of a number of nodes that interconnected with each other through mathematical functions. According to Krenker A. et al)Andrej Krenker ، (2011the basic element of any ANN is a neuron, which is designed based on the neuron in a biological neural network. This neuron is structured as shown in Figure 2.

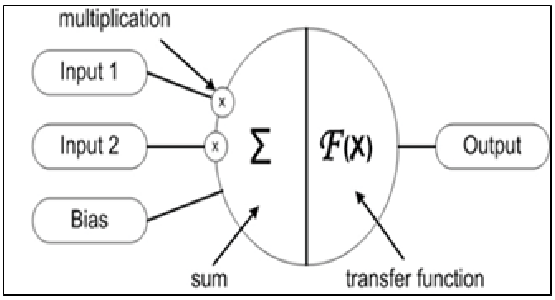

Figure 2. The structure of the artificial neuron (Andrej Krenker, 2011)

Each neuron operates by receiving inputs, which are either the system inputs or the output from other neurons that are connected with this neuron. These inputs are weighted Each neuron operates by receiving inputs, which are either the system inputs or the output from other neurons that are connected with this neuron. These inputs are weighted individually and the neuron sums it with each other and with the bias, and then the result of this summation is processed through transfer function.

In addition, According to Haykin S.)Haykin ، (2008 , the learning concept of ANN is categorized as the following:

Supervised learning: In this learning methodology of neural network , the system is trained by providing it with desired behavior (output data) for a set of specific inputs.

Unsupervised learning: This type of learning algorithm does not require providing target output. it may seems difficult to illustrate , however it can be simplified as an algorithm that aims to find a pattern in given input data , which can be used for decision making , prediction and so on (Ghahramani, 2004).

Reinforcement learning: This type of learning algorithm aims on interacting with its environment “to learn to act in a way that maximizes the future rewards it receives (or minimizes the punishments) over its lifetime” (Ghahramani, 2004).

2.3 Backpropagation Neural Network

Backpropagation neural network (BNN) is a one of the most popular supervised ANN, which is, as the name implies, uses the backpropagation algorithm concept for learning. It is developed in 1970 to solve the limitation of neural network (NN) algorithm, which failed to address XOR issue (Shihab, 2006). According to shihab K. the BNN is basically consist of a small pieces that interconnect together to solve complex issues (Shihab, 2006). It is a feed forward with a structure of Multi-layer, which is learning based on error back feeding (Jing Li, 2012).

According to (RashmiAmardeep, 2017) , BNN is considered as a type of learning and training algorithm rather than being a type of neural network. In order to train this network, it is required to provide the BNN with output data for specific input. After training the network, it will be ready to recognize any input pattern based on the pattern of training data (RashmiAmardeep, 2017). Typically, the structure of a standard backpropagation network consist of three layers: input, hidden layer and output layer (Mutasem Alsmadi, 2009). This structure is shown in Figure 3.

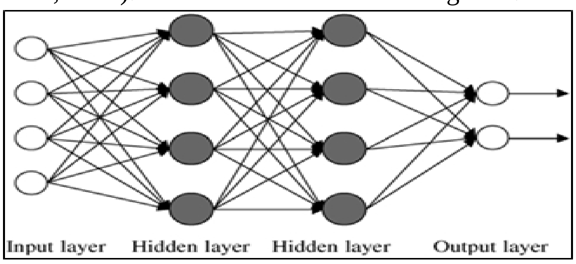

Figure 1 The structure of Backpropagation network (Mutasem Alsmadi, 2009)

The learning algorithm of backpropagation is essentially based on the theory of error –correction-learning concept “which uses the error function in order to modify the connection weights to gradually reduce the error (Alaeldin Suliman, 2015).

In another word, the network modifies the connections between its layers' neurons by changing the weights that related to each of these neuron functions as illustrated in the previous section. The weight changing aims to minimize the differences between current output of the network and desired target output. Many applications can be used to build and design a neural network with backpropagation methodology such as Simbrain and Matlab.

2.4 NeMo

The core building block in NeMo is called Neural Module (NM). A Neural Nodule represents a logical part of a neural network such as a language model, an encoder, a decoder, a data augmentation algorithm, a loss function, or other sets of layers and functions. As the primary abstraction in NeMo, NMs form the basis for describing a model and the process by which that model is trained. Formally, a Neural Module is a component that computes a set of typed outputs given a set of typed inputs. Inputs and outputs are collections of multidimensional tensors. In the same way that a programmer in an object-oriented language can choose at what level of granularity to define an object, a NeMo user can choose the level of granularity of a Neural Module. A basic rule is that inputs and outputs should "make sense" to expose via an interface. This suggests that a Neural Module is not typically a single neural network layer, but rather a collection of connected layers that "do something useful" such as an encoder, a concatenation operation, a loss function, or a data augmentation.

NeMo Core

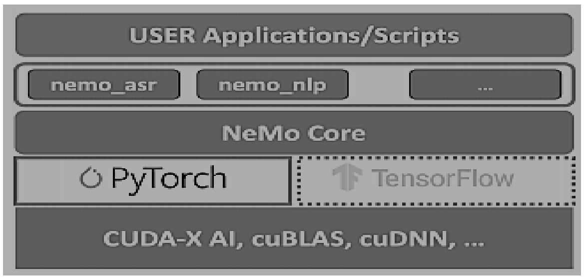

Figure 4: NeMo is a framework-agnostic toolkit which serves as abstraction level between application and DL frameworks

NeMo allows users to create callbacks for routines performed during training such as evaluation, logging, and performance monitoring.

III. METHODOLOGY

The aim of this work is to test and investigate the ability of BNN via NeMo toolkit to distinguish and recognize the pattern of python programming language. This is conducted by applying the algorithm of pattern recognition, which is presented by Sharma and Kaur (Priyanka Sharma, 2013); see Figure 1. Firstly, the data are collected from the most common syntax codes of python language. Then these codes are converted into a form of binary array, from which the features of potential pattern are selected. This is follow by selecting BNN as the model used for pattern recognition. This model is then trained using the collected data. Finally, the trained BNN model is evaluated by applying three tests. In the first test, the evaluation is conducted by applying the proposed BNN with ten arbitrary python syntax codes to check how it behaves. While in the second test, the system is checked using ten words and sentences to investigate its ability to recognize non-python pattern. Eventually, the third test check the confusion behavior of the system when it is applied with ten confusion data. In addition, another objective of this work is to check the ability of NeMo toolkit to represent analyzable, measurable results. This is conducted by observing the results of each of above results numerically.

3.1 Collected Data

The input data for this work are collected from the most used addition of two number python program programming syntax codes that distinguish python programming language from other languages. Instance for these codes can be as the following:

```
a = 1
b= 2
sum = a + b
print ("sum ",sum)
```

Table 1. Sample of input data representation

Tim	a	b	c	d	e	f	g	h	I	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	;	(	)	“	“	=	+	,	1	2
a=1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0 1	0	1	0	0	1	0
b=2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
Sum=a+b.	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
Print("sum:" ,sum)	1	0	0	0	0	0	0	0	1	0	0	0	1	1	0	1	0	1	1	1	1	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0

IV. DESIGNING THE BACKPROPAGATION NEURAL NETWORK

As presented previously, the proposed neural network is designed and implemented using NeMo toolkit. This is conducted by creating a neural network with 36 input neurons in in the first layer, 16 neurons in the hidden layer, and 2 neurons in the output layer.

The 36 input neurons represent the signs for python programming code, while the 2 neurons in output layers represent the indication part of the model that detect whether the inputs pattern is python programming language or not . This is shown in Figure 4 and 5

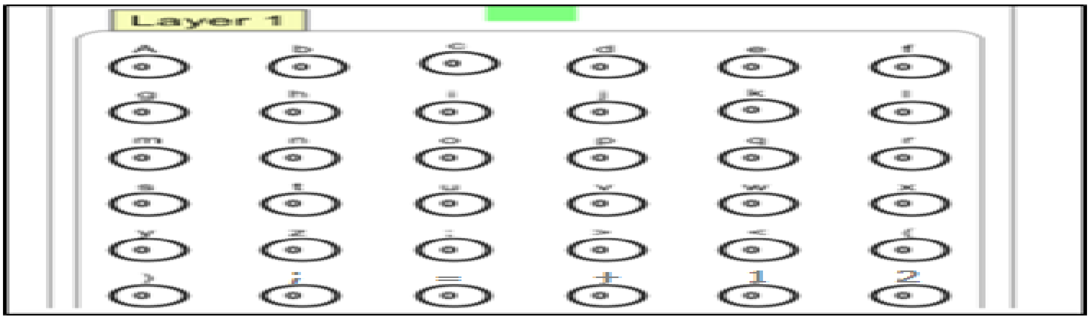

Figure 4. The first layer of the network

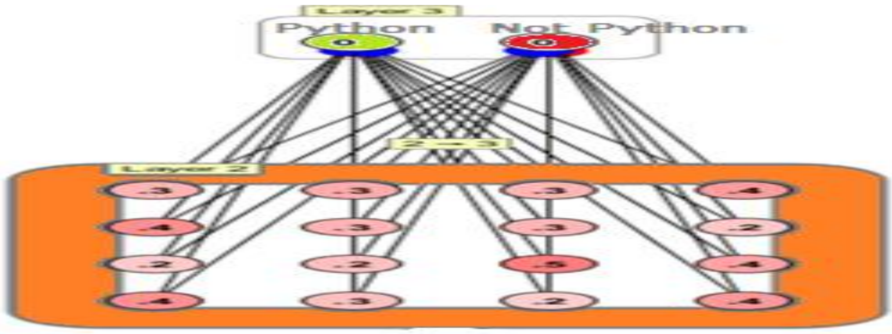

Figure 5. The second and the third layer of the neural network

It can be noticed that the NeMo platform provides the ability to illustrate the interconnection between these layers.

V. TRAINING THE MODEL

The proposed NeMo back propagation neural network (SBNN) is trained in this work by applying the network with 1500 training inputs and 1500 target outputs. As presented previously, these inputs are structured in a form of zero-one, which are imported into the system in CSV file. A sample of these inputs is shown in Figure 6.

The training inputs can be classified into two groups. First group is a data set of python codes, while the second group is a data set of non-python words pattern.

Table 2: Training input in .CSV file

Input data

#	Neuron_1	Neuron_2	Neuron_3	Neuron_4	Neuron_5	Neuron_6	Neuron_7	Neuron_8	Neuron_9	Neuron_
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
3	0.0	0.0	1.0	1.0	11.0	0.0	0.0	0.0	11.0	0.0
4	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	1.0	0.0	1.0	0.0	11.0	0.0	1.0	0.0	1.0	0.0
6	1.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	11.0	0.0
7	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
8	11.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0
10	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0
11	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0
12	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0
13	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
14	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
16	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
18	1.0	0.0	1.0	1.0	11.0	0.0	0.0	0.0	11.0	0.0
19	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20	1.0	0.0	1.0	0.0	11.0	0.0	1.0	0.0	1.0	0.0
21	1.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	11.0	0.0
22	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
23	11.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
24	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0

Target data

#	Neuron_53	Neuron_54
1	0.0	0.0
2	1.0	0.0
3	1.0	0.0
4	1.0	0.0
5	1.0	0.0
6	1.0	0.0
7	1.0	0.0
8	1.0	0.0
9	1.0	0.0
10	1.0	0.0
11	1.0	0.0
12	1.0	0.0
13	1.0	0.0
14	1.0	0.0
15	1.0	0.0
16	1.0	0.0
17	1.0	0.0
18	1.0	0.0
19	1.0	0.0
20	1.0	0.0
21	1.0	0.0
22	1.0	0.0
23	1.0	0.0
24	1.0	0.0
25	1.0	0.0

Let's train the Neural Network for 1500 iterations and see what happens. Looking at the loss per iteration graph below,

Figure 6: Loss per Iteration rate

IV. ANALYSIS AND DISCUSSION

According to the results presented in previous section, the behavior of SBNN model shows high accuracy of pattern recognition. This ability to measure the accuracy and certainty of the system can be noticed from the results, for instance when the pattern is more likely python, the network gives more probability for python than not python in the output layer. The network shows this probability in the form of a range of number between 0 and 1. When 0 represent 0% while 1 represent 100%. This range and certainty depends on the quantity and the quality of the data used in the system training.

Furthermore, the results also show the high numerical and graphical ability of NeMo toolkit for analysis and study this can be seen in the behaviour of the second layer, see Figure 5. This gives an opportunity to study the model in more details for future works.

VII. CONCLUSION

By growing the diversity of software applications, the programming languages that are used to develop these applications upturn too. Consequently, taking an automated tool to differentiate these programming languages would be very useful for developers and scholars. In this paper we demonstrated a pattern recognition model using back propagation neural network via NeMo toolkit to recognize python programming language codes. This model success to identify the patterns of different inputs with high accuracy. The result also shows a high ability of NeMo to provide numerical and graphical results for both research study and analysis. For future work, it is recommended to increase training data and perform more tests on the model. In addition, it is suggested to test the proposed pattern neural network via matlab toolkit and using the matlab model to conduct a comparative evaluation study with the current NeMo model.

REFERENCES:

1. A. B. Philip Mayer, “An Empirical Analysis of the Utilization of Multiple,” in 19th International Conference on Evaluation and Assessment in Software, April 2015.
2. R. Montenegro, “Source Code Classification Using Deep Learning,” 30 8 2016. [Online]. Available: http://blog.aylien.com/source-code-classification-using-deep-learning/. [Accessed 7 8 2019].
3. M. S. J. D. G. S. Jyotiska Nath Khasnabish, “Detecting Programming Language from Source Code Using Bayesian Learning Techniques,” Springer International Publishing Switzerland 2014, p. 513–522, 2014.
4. C. Sargur N. Srihari, “pattern recognition,” London, Chapman, 1993, pp. 1034-1041.
5. E. SALIBA, “An overview of Pattern Recognition,” University of Burgundy, Antonine University, pp. 1-7, 2014.
6. D. B. ,. T.-h. K. Jayanta Kumar Basu, “Use of Artificial Neural Network in Pattern Recognition,” International Journal of Software Engineering and I International Journal of Software Engineering and Its Applications, vol. 4, no. 2, pp. 23-34, 2010.
7. M. K. Priyanka Sharma, “Classification in Pattern Recognition: A Review,” International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, no. 4, pp. 298-306, 2013.
8. S. W. Smith, “Chapter 26: Neural Networks (and more!),” in The Scientist and Engineer's Guide to digital signal processing, California, California Technical Pub; 1st edition (1997), 1997, p. 626.
9. K. Gurney, An introduction to neural networks, london, new york: UCL Press, 1997.
10. J. B.,. A. K. Andrej Krenker, “Introduction to the Artificial Neural Networks,” in Artifital neural network, In Tech, 2011, p. 362.
11. S. Haykin, Neural Networks and Learning Machines Third Edition, Hamilton, Ontario, Canada: Pearson, 2008.
12. Z. Ghahramani, “Unsupervised Learning,” Springer-Verlag Berlin Heidelberg, pp. 72-112, 2004.
13. K. Shihab, “A Backpropagation Neural Network for Computer Network Security,” Journal of Computer Science, vol. 2, no. 9, pp. 710-715, 2006.
14. J.-h. C. J.-y. S. F. H. Jing Li, “Brief Introduction of Back Propagation (BP) Neural,” springer, vol. 2, pp. 553-558, 2012.
15. K. T. RashmiAmardeep, “Training Feed forward Neural Network With Backpropogation Algorithm,” International Journal Of Engineering And Computer Science, vol. 6, no. 1, pp. 19860-19866, 2017.
16. K. O. . S. A. M. N. Mutasem Alsmadi, “Back Propagation Algorithm: The Best Algorithm among the Multi-layer Perceptron Algorithm,” International Journal of Computer Science and Network Security, vol. 9, no. 4, pp. 378-383, 2009.
17. Y. Z. Alaeldin Suliman, “A Review on Back-Propagation Neural Networks in the Application of Remote Sensing Image Classification,” Journal of Earth Science and Engineering, vol. 5, pp. 52-65, 2015.
18. j. y. Zachary Tosi, “Simbrain 3.0: A flexible, visually-oriented neural network simulator,” Elsevier journal neural network, vol. 83, pp. 1-10, 2016.
19. W.-L. L. ,. L. L. Tung-Hsu Hou, “Intelligent remote monitoring and diagnosis of manufacturing processes using an integrated approach of neural networks and rough sets,” Journal of Intelligent Manufacturing , vol. 14, no. 2, pp. 239-253, 2003.