Digital Cryptography Implementation using Neurocomputational Model with Autoencoder Architecture

Francisco Quinga Socasi[a], Ronny Velastegui[b], Luis Zhinin-Vera[c], Rafael Valencia-Ramos[d], Francisco Ortega-Zamorano[e] and Oscar Chang[f]

School of Mathematical and Computational Sciences, Yachay Tech University, 100650, Urcuqui, Ecuador

Keywords: Cryptography, Artificial Neural Network, Autoencoder, ASCII Characters.

Abstract: An Autoencoder is an artificial neural network used for unsupervised learning and for dimensionality reduction. In this work, an Autoencoder has been used to encrypt and decrypt digital information. So, it is implemented to code and decode characters represented in an 8-bit format, which corresponds to the size of ASCII representation. The Back-propagation algorithm has been used in order to perform the learning process with two different variant depends on when the discretization procedure is carried out, during (model I) or after (model II) the learning phase. Several tests were conducted to determine the best Autoencoder architectures to encrypt and decrypt, taking into account that a good encrypt method corresponds to a process that generate a new code with uniqueness and a good decrypt method successfully recovers the input data. A network that obtains a 100% in the two process is considered a good digital cryptography implementation. Some of the proposed architecture obtain a 100% in the processes to encrypt 52 ASCII characters (Letter characters) and 95 ASCII characters (printable characters), recovering all the data.

1 INTRODUCTION

Cryptography is a science that allows to write messages so that only authorized people can read them. The goal of cryptography is keep the integrity, confidentiality, and authenticity of the information. Messages are encrypted and decrypted around a key. This key is considered as the unique way to read an encrypted message. There are many techniques to encrypt information that are classified depending on what type of key is used: secret key, public key and hash function. However, in all the cases the message that we want to encrypt is called plain text, and the encrypted text is denominated cipher text (Stallings, 1998).

Advanced Encryption Standard (AES) is one of the most used encryption algorithms today. Since 2002, AES is a worldwide standard for information encryption/decryption. This algorithm allows to encrypt blocks of information of 128 bits, using a symmetric key of 128, 192 or 256 bits (Forouzan, 2007).

Artificial neural networks could be defined as networks massively interconnected in parallel of simple elements and with hierarchical organization, which try to interact with the objects of the real world in the same way as the biological nervous system does (Matich, 2001).

An Autoencoder is a feed-forward multilayer neural network that learns to reproduce the same information that it receives input, in the output. So, is a multilayer neural network consisting of: an input layer, an output layer (both with an equal number of neurons), and one or more intermediate layers, also called hidden layers. At first glance, it seems that it is a useless network since it simply replicates the information it receives. But, in reality, the key lies in the internal structure and general architecture of the said network (Tan, 1998).

In recent years, investigations on neural networks linked to cryptography have increased considerably. Due to the nature of artificial neural networks, and to their great combination of weights and connections, it is possible to hide and compress information in their structure (Lonkar and Charniya, 2014).

So, neural networks coupled with cryptography give us a novel approach to encrypt and decrypt information.

In this paper, an Autoencoder to encrypt and decrypt digital information has been used. This infor-

[a] https://orcid.org/0000-0003-3213-4460
[b] https://orcid.org/0000-0001-8628-9930
[c] https://orcid.org/0000-0002-6505-614X
[d] https://orcid.org/0000-0002-1036-1817
[e] https://orcid.org/0000-0002-4397-2905
[f] https://orcid.org/0000-0002-4336-7545

mation is represented in an 8-bit format, which corresponds to an ASCII (American Standard Code for Information Interchange) character.

Because the optimal architecture of the neural networks depends in its great majority of its specific application (Rojas, 1996), we will compare two proposed models of architectures and try to find the best it as a model of encryption / decryption.

For this, we will perform a series of tests in which we will vary some parameters such as: number of internal neurons, among others. And we will observe the accuracy of the entire network after each test. And so, we will choose the network with the greatest accuracy.

2 RELATED WORK

Neural networks can accurately identify a nonlinear system model from the inputs and outputs of a complex system, and do not need to know the exact relationship between inputs and outputs (Charniya, 2013). All this makes the use of an ANN viable, in the process of encryption and decryption of data. Artificial Neural Networks offer a very powerful and general framework for representing the nonlinear mapping of several input variables to several output variables. Based on this concept, an encryption system by using a key that changes permanently was developed. In Volna's work, the topology is very important issue to achieve a correct function of the system, therefore a multilayer topology was implemented, considered the more indicated topology in this case. Also for the encryption and decryption process, it was carried out using a Back-propagation; technique compatible with the topology (Volna et al., 2012).

Neural networks can be used to generate common secret key (Jogdand, 2011). The neural cryptography, exist two networks that receive an identical input vector, generate an output bit and are trained based on the output bit. Both networks and their weight vectors exhibit a novel phenomenon, where the networks synchronize to a state with identical time-dependent weights. The generated secret key over a public channel is used for encrypting and decrypting the information being sent on the channel.

A watermarking technique to hides information in images to diminish copyright violations and falsification is proposed (Wang et al., 2006). Basically, it embeds a little image that represent a signature into another image. This paper uses techniques as human visual system (HVS) and discrete wavelet transform (DWT) to decompose the host image in L-levels. This is the reason for the Wavelet Domain in the tittle. The method is embed the watermark into the wavelet coefficients chosen by HVS (brightness, weight factor). This uses neural network to memorize the relationship between the watermark W and the wavelet coefficients I. The topology is a 8, 5 and 1 layer for input, hidden and output layer respectively. When the network is trained it is capable of recover the watermark image. The experiment introduces different ranges of noise into the hos image and see the capability of recover the watermark image.

The application of interacting neural networks for key exchange over a public channel is showed (Kinzel and Kanter, 2016). It seems that two neural networks mutely trained, achieve a synchronization state where its time dependent synaptic weights are exactly the same. Neural cryptography uses a topology called tree parity machine which consist of one output neuron, K hidden neurons and K*N input neurons. The hidden values are equal to the sign function of dot product of input and weights while the output value is the multiplication of hidden values. The training process is carried out comparing the outputs of the corresponding tree parity machines of two partners A and B. It has not been proved that no exits an algorithm for success attack, but this approach is very hard to crack by brute force. Even though an attacker knows the input/output relation and knows the algorithm, he is not able to recover the secret common key that A and B uses for encryption. Neural networks are the unique algorithm for key generation over a public channel that is not based on number theory. Its main advantages over traditional approaches are: it simple, low computations for training and a new key is generated for each message exchange.

Likewise, the applications of mutual learning neural networks that get synchronization of their time dependent weights is showed (Klein et al., 2005). It suggests that synchronization is novel approach for the generation of a secure cryptographic secret-key using a public channel. For a further insights over the synchronization, the process is analytically described using statistical physics methods. This works describes the learning process in a simple network, where two perceptrons receive a common and change their weights according to their mutual output, and the learning process in a tree parity machines. In addition, a new technique that combines neural networks with chaos synchronization. This seems to be the most secure against attackers. Finally, this works explains the different kind of attacker techniques and suggest that using a big enough weight space the systems becomes more secure.

Another field in which neural networks have been applied together with cryptography is Steganalysis.

Artificial Neural Network is utilized as the classifier (Shi et al., 2005). This paper concludes that ANN performs better in Steganalysis than Bayes classifier due to its powerful learning capability. Steganalysis is the art and science to detect whether a given medium has hidden message in it.

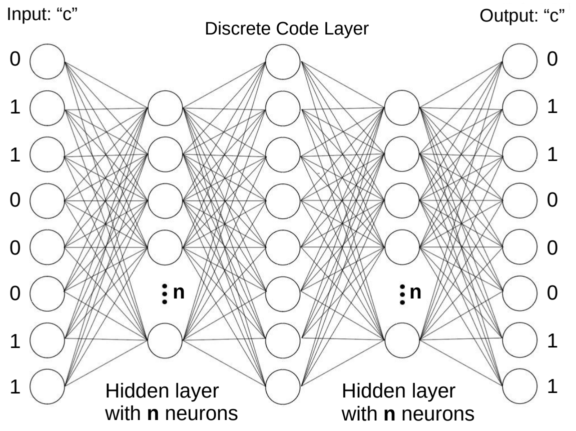
Pseudo random number generator is another approach for neural networks in cryptography. Randomness increases the security of cryptosystem. The result of many tests claim that the resulting generator is well suited for practical implementation of efficient stream cipher cryptosystems (Cheng and Chan, 1998). The pseudo random number generator takes the advantage of multi-layer perception (MLP) neural networks. In over-fitting process, the network will not be able to predict the input pattern when receiving unknown input patterns and will give unpredictable results (Karras and Zorkadis, 2003). MLP neural network can be used as strong independent random generator and can also be used as a method of strengthening the current existing generators by taking the pseudo random numbers output by linear computational generators as input to the neural networks (Hadke and Kale, 2016).

3 AUTOENCODER FOR ENCRYPTION AND DECRYPTION OF ASCII CODE

3.1 Autoencoder Architecture

The architecture of an artificial neural network with Autoencoder is composed by a first layer in which the input data is presented, and an output layer in which the presented data is recovered. Both layers have to be implemented with equal number of neural units, i.e.neurons, and one or more intermediate layers, also called hidden layers (Buduma, 2015). In addition, in order to use the Autoencoder for encryption / encryption purposes, the architecture must be symmetric. Also, the central inner layer must be of discrete output, and the same size as the input and output layers. This architecture is shown in the Figure 1.

When entering information in the network, it must find a way to transfer it through each hidden layer. Once the information reaches the central hidden layer, we obtain a discrete exit of the same size as the entrance. So at this point, we have made an encryption of our information. After the information is in the central layer (where the information is currently encrypted), this information continues through the rest of the network, and in the end, we will obtain our original information. So, we have made a decryption of information. Therefore, once trained the network, we can divide it into two parts, the first is the encryptor, and the second is the decryptor.


Input: "c"
Discrete Code Layer
Output: "c"
0
1
1
0
0
0
1
1
n
n
Hidden layer with **n** neurons
Hidden layer with **n** neurons
0
1
1
0
0
0
1
1
Figure 1: Autoencoder architecture of the proposed method.

3.2 Back-Propagation Algorithm

The Back-Propagation algorithm is a supervised learning method for training multilayer artificial neural networks, and even if the algorithm is very well known, we summarize in this section the main equations in relationship to the implementation of the Back-Propagation algorithm, as they are important in order to understand the current work.

Let's consider a neural network architecture comprising several hidden layers. If we consider the neurons belonging to a hidden or output layer, the activation of these units, denoted by y_i, can be written as:

$$y_i = g\left(\sum_{j=1}^{L} w_{ij} \cdot s_j\right) = g(h) \, , \tag{1}$$

where w_{ij} are the synaptic weights between neuron i in the current layer and the neurons of the previous layer with activation s_j. In the previous equation, we have introduced h as the synaptic potential of a neuron. The activation function used, g, is the logistic function given by the following equation:

$$g(x) = \frac{1}{1+e^{-\beta x}}, \tag{2}$$

The objective of the BP supervised learning algorithm is to minimize the difference between given outputs (targets) for a set of input data and the output of the network. This error depends on the values of the synaptic weights, and so these should be adjusted in order to minimize the error. The error function computed for all output neurons can be defined as:

$$E = \frac{1}{2}\sum_{k=1}^{p}\sum_{i=1}^{M}(z_i(k) - y_i(k))^2, \tag{3}$$

where the first sum is on the p patterns of the data set and the second sum is on the M output neurons. $z_i(k)$ is the target value for output neuron i for pattern k, and $y_i(k)$ is the corresponding response output of the network. By using the method of *gradient descent*, the BP attempts to minimize this error in an iterative process by updating the synaptic weights upon the presentation of a given pattern. The synaptic weights between two last layers of neurons are updated as:

$$\Delta w_{ij}(k) = -\eta \frac{\partial E}{\partial w_{ij}(k)} = \eta[z_i(k) - y_i(k)]g_i'(h_i)s_j(k), \tag{4}$$

where η is the learning rate that has to be set in advance (a parameter of the algorithm), g' is the derivative of the sigmoid function and h is the synaptic potential previously defined, while the rest of the weights are modified according to similar equations by the introduction of a set of values called the “deltas” (δ), that propagate the error from the last layer into the inner ones, that are computed according to Eqs. 5 and 6.

The delta values for the neurons of the last of the N hidden layers are computed as:

$$\delta_j^N = (S_j^N)'[z_j - S_j^N], \tag{5}$$

The delta values for the rest of the hidden layer neurons are computed according to:

$$\delta_j^l = (S_j^l)' \sum w_{ij}\delta_i^{l+1}, \tag{6}$$

3.3 Discretization Methods of the Middle Layer

Two different methods of discretization of the middle layer have be implemented in order to analyze the better implementation for the autoencoder with encryption use.

3.3.1 Model I

In Model I, the learning algorithm Back-Propagation is used, but with the difference that the output of the central layer is always given in discrete values (by rounding).

3.3.2 Model II

In Model II, the normal Back-Propagation learning algorithm is used. So, during training, the output of the central layer is always given in continuous values. The discretization of the central layer is done only once the training has been completed.

4 EXPERIMENTAL STUDY

The architecture employed for the autoencoder methods has been with 8 input neurons in order to represent ASCII characters, n neurons in the first hidden layer, 8 neurons in the second hidden layer to represent the codification of the data, n neurons in the third hidden layer, and finally 8 neurons in the output layer to recovered the initial data. The architecture is shown in Figure 1, taking into account, that n has been changed to analyze the best architecture.

Cryptographic tests using two sets of input data have been performed. The first input set contains 52 patterns, that is, 52 ASCII characters (letter characters). While the second input set is of a larger size, since it consists of 95 ASCII characters (printable characters).

Table 1 shows the results in terms of Accuracy and Uniqueness of the resultant binary code for Model I, using an input data set of 52 patterns. The first column correspond to the number (n) of neurons in the second and fourth layer, for purposes of this investigation, n can take one of these values: 7, 8, 9, 10. The “Beta” field is the gain value of the transfer function used, in this research it is the sigmoid function. For this investigation, the possible values that “Beta” can take are: 5, 10, 15, 20, 25. The field “Min MSE” corresponds to the minimum mean squared error reached during the training process of this network. Then, “Epoch” corresponds to the epoch number in which the value specified in “Min MSE” was reached first. Then, the field “Accuracy” corresponds to the accuracy of the architecture in the recovery of the 52 patterns. Therefore, this field “Accuracy” is the most important in our table, since only architectures that achieve a 100% of “Accuracy” will be considered valid architectures for cryptographic purposes. Architectures that do not achieve an Accuracy of 100% will be called simply invalid. Finally, the field “Uniq.Cod.Layer” is derived directly from the field “Accuracy”, and means the percentage of uniqueness of outputs with respect to the input.

Now, once all the fields in Table 1 have been defined, it is time to interpret their contents. As we already know, we are only interested in valid archi-

Table 1: Results of the architectures, 52 inputs, Model I.

Arch.	Beta	Min MSE	Epoch	Acc. (%)	Uniq. Cod. Layer (%)
Arch. 1 (n=10)	5	0.1979	680	78.8%	80.7%
	10	0.0426	749	92.3%	98%
	15	**0.0238**	**385**	**100%**	**100%**
	20	0.0657	1000	96.1%	98%
	25	0.5044	375	44.2%	46.1%
Arch.2 (n=9)	5	0.2338	742	67.3%	82.6%
	10	0.1263	123	80.7%	86.5%
	15	0.0350	430	96.1%	96.1%
	20	0.0842	125	88.4%	96.1%
	25	0.4765	48	51.9%	69.2%
Arch. 3 (n=8)	5	0.2863	747	59.6%	71.1%
	10	0.0172	325	94.2%	98%
	15	0.0541	137	96.1%	96.1%
	20	0.2957	109	63.4%	76.9%
	25	0.7087	102	34.6%	46.1%
Arch. 4 (n=7)	5	0.2937	574	55.7%	71.1%
	10	0.0567	498	88.4%	92.3%
	15	0.3983	442	55.7%	59.6%
	20	0.3840	68	51.9%	63.4%
	25	0.6497	90	34.6%	44.2%

Table 2: Results of the architectures, 95 inputs, Model I.

Arch.	Beta	Min MSE	Epoch	Acc. (%)	Uniq. Cod. Layer (%)
Arch.1 (n=10)	5	0.2572	727	50.5%	66.3%
	10	0.3064	156	63.1%	70.5%
	15	0.3151	54	56.8%	73.6%
	20	0.4340	84	51.5%	68.4%
	25	0.9769	87	22.1%	35.7%
Arch. 2 (n=9)	5	0.3410	640	46.3%	49.4%
	10	0.1566	258	70.5%	80.0%
	15	0.2699	146	66.3%	72.6%
	20	0.7556	25	25.2%	42.1%
	25	1.1362	88	14.7%	20.0%
Arch. 3 (n=8)	5	0.4167	999	41.0%	46.3%
	10	0.3639	133	47.3%	61.0%
	15	0.4116	61	48.4%	54.7%
	20	0.6323	83	33.6%	36.8%
	25	0.9375	156	18.9%	29.4%
Arch.4 (n=7)	5	0.4680	787	35.7%	43.1%
	10	0.4554	60	37.8%	68.4%
	15	0.5182	461	37.8%	43.1%
	20	0.7462	306	32.6%	40.0%
	25	1.1851	76	11.5%	20.0%

tectures, that is, those that achieve a 100% accuracy. Thus, in Table 1, we observe that only architecture 1 with "Beta" = 15, is a valid architecture, because the 100% accuracy was obtained in the 385 epoch.

Table 2 shows the results in terms of Accuracy and Uniqueness of the resultant binary code for Model I, using an input data set of 95 patterns. Consider that the columns in Table 2 have the same meaning as the columns in Table 1.

In Table 2, we can see that in no architecture was it possible to obtain an Accuracy of 100%. Also, it can even be noted that these values of accuracy are much lower than those in Table 1. Therefore, there is no valid architecture in Table 2.

Table 3 summarizes the results in terms of Accuracy and Uniqueness of the resultant binary code for Model II, using an input data set of 52 patterns. Consider that the columns in Table 3 have the same meaning as the columns in Table 1.

In Table 3, we can see that in 4 architectures it was possible to obtain an accuracy of 100%. In other words, using Model II, with 52 inputs, we obtained 4 valid architectures for cryptographic purposes. Two of them with n = 10, one of them with n = 9, and the last with n = 8. And all of them with a "Beta" between 15 or 20.

Table 4 shows the results in terms of Accuracy and Uniqueness of the resultant binary code for Model II, using an input data set of 95 patterns. Consider that the columns in Table 4 have the same meaning as the columns in Table 1.

In Table 4, we can see that in 3 architectures it was possible to obtain an accuracy of 100%. In other words, using Model II, with 95 inputs, we obtained 3 valid architectures for cryptographic purposes. One with n = 10, other with n = 9, and the last with n = 8. And all of them with a "Beta" = 15.

Figure 2 show the evolution of Accuracy with respect to the number of epoch elapsed during training of Model II, with an input data set of 52 patterns.

With an input data set of 52 patterns, we notice that four architectures of Model II are valid (this we had already noted in Table 3), that is, they reach an accuracy of 100%. Additionally, in this 4 valid architectures, the maximum accuracy is reached before the 300 epoch.

Figure 3 has the same structure of Figure 2. But in this case the results represented were obtained with an input data set of 95 patterns using Model II.

We notice that three architectures of Model II are valid (this we had already noted in Table 3), that is, they reach an accuracy of 100%. Additionally, in this 3 valid architectures the maximum accuracy is reached before the 300 epoch.

5 DISCUSSION AND CONCLUSIONS

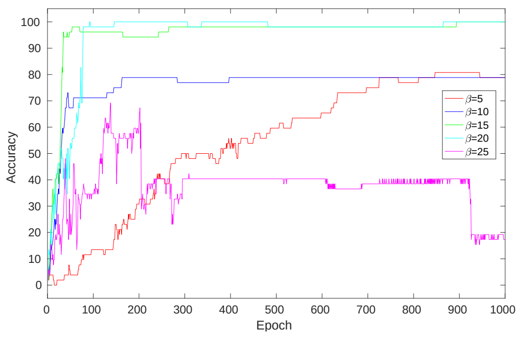
A new method to encrypt and decrypt digital information by autoencoder architecture of a neural network model has been analyzed. Two architecture models with different learning algorithms were proposed. Between both models, clearly the best one for cryptographic purposes is the Model II. Since that model produced more valid architectures (with 100% of accuracy). This difference between Models I and II, is due to the learning algorithm used. In the first model a small variation of the back-propagation is used, in

Table 3: Results of the architectures, 52 inputs, Model II.

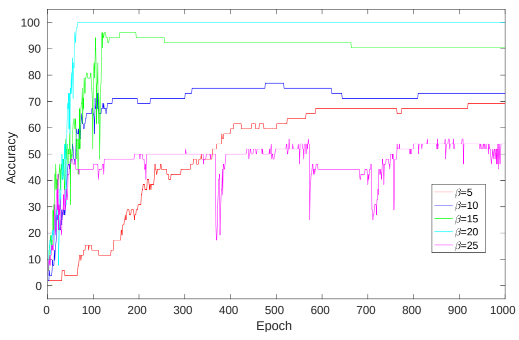
Arch.	**Beta**	**Min MSE**	**Epoch**	**Acc. (%)**	**Uniq. Cod. Layer (%)**
Arch. 1 (n=10)	5	0.0019	1000	78.84%	94.23%
	10	3.97e-04	1000	78.84%	92.3%
	15	**1.38e-04**	**1000**	**100%**	**100%**
	20	**8.35e-05**	**1000**	**100%**	**100%**
	25	0.255	138	69.23%	73.07%
Arch.2 (n=9)	5	0.0027	1000	69.23%	84.61%
	10	5.27e-04	1000	73.07%	96.15%
	15	1.32e-04	1000	90.38%	94.23%
	20	**1.39e-04**	**1000**	**100%**	**100%**
	25	0.3876	548	53.84%	61.53%
Arch. 3 (n=8)	5	0.004	1000	63.46%	86.53%
	10	3.00e-04	1000	71.15%	90.38%
	15	1.36e-04	1000	92.30%	98.07%
	20	**9.65e-05**	**1000**	**100%**	**100%**
	25	0.4008	196	59.61%	65.38%
Arch.4 (n=7)	5	0.0055	1000	65.38%	92.30%
	10	8.78e-04	1000	65.38%	88.46%
	15	0.0015	1000	86.53%	92.30%
	20	1.15e-04	1000	96.15%	98.07%
	25	0.3412	159	59.61%	65.38%

Table 4: Results of the architectures, 95 inputs, Model II.

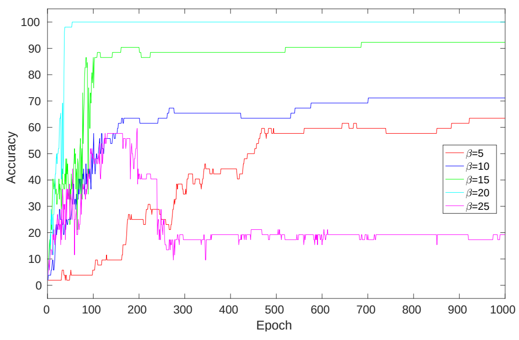
Arch.	**Beta**	**Min MSE**	**Epoch**	**Acc. (%)**	**Uniq. Cod. Layer (%)**
Arch. 1) (n=10)	5	0.011	1000	61.05%	80%
	10	2.56e-04	1000	75.78%	89.47%
	15	**7.60e-05**	**1000**	**100%**	**100%**
	20	0.0837	209	89.47%	89.47%
	25	0.7764	132	26.31%	32.63%
Arch. 2 (n=9)	5	0.013	1000	49.47%	75.78%
	10	2.73e-05	1000	78.94%	88.42%
	15	**1.35e-04**	**1000**	**100%**	**100%**
	20	5.04e-05	1000	97.89%	100%
	25	1.1049	34	28.42%	35.78%
Arch. 3 (n=8)	5	0.0017	1000	48.42%	73.68%
	10	2.40e-04	1000	67.36%	76.84%
	15	**9.23e-05**	**1000**	**100%**	**100%**
	20	0.0636	174	91.57%	91.57%
	25	0.7255	928	33.68%	35.78%
Arch. 4 (n=7)	5	0.0413	999	38.94%	69.47%
	10	0.1268	1000	66.31%	76.84%
	15	1.46e-04	1000	91.57%	96.84%
	20	0.1685	1000	78.94%	80%
	25	0.8556	26	28.42%	42.1%



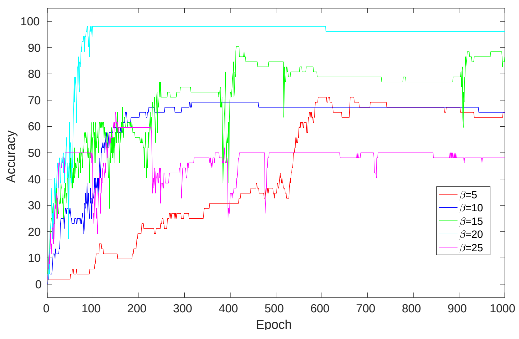
(a) Architecture 1 (n=10)



(b) Architecture 2 (n=9)

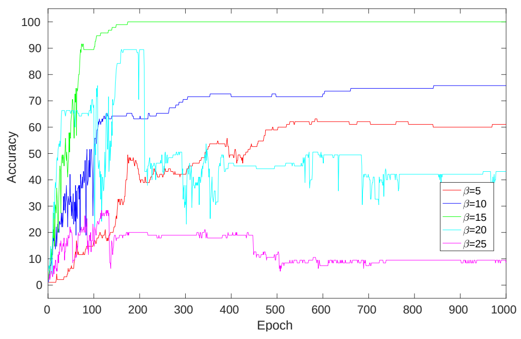


(c) Architecture 3 (n=8)

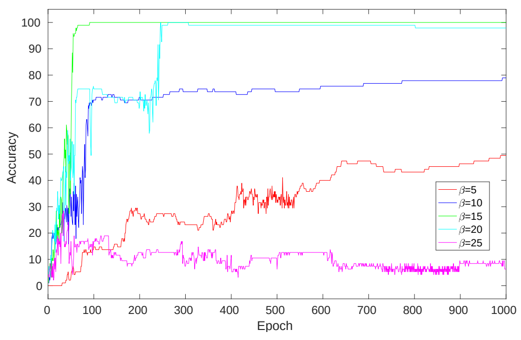


(d) Architecture 4 (n=7)

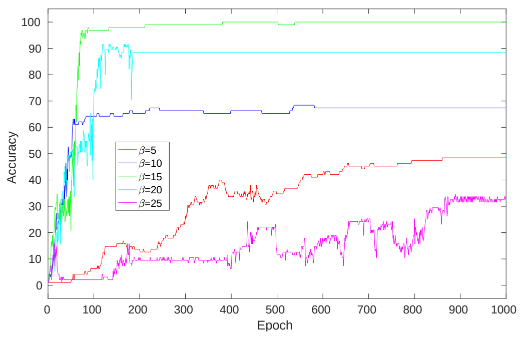
Figure 2: Accuracy of Model II for every architecture and 52 inputs.



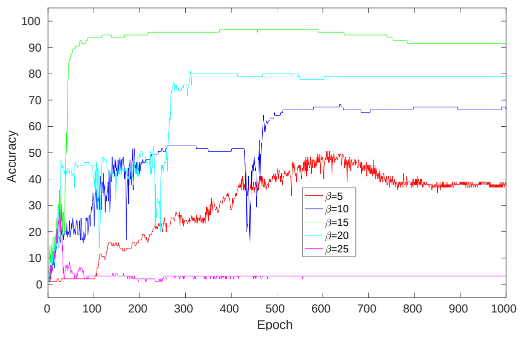
(a) Architecture 1 (n=10)



(b) Architecture 2 (n=9)



(c) Architecture 3 (n=8)



(d) Architecture 4 (n=7)

Figure 3: Accuracy of Model II for every architecture and 95 inputs.

which a discretization (rounding) of the outputs of the intermediate layer is carried out during the training. This rounding produces a loss of uniqueness and therefore loss of precision when retrieving patterns.

Once we identify the Model II, as the best model. It was necessary to choose the best architecture of said model. This architecture is the one that recovers all the patterns (accuracy 100%), and that has the smallest number (n) of neurons.

In Table 4, there are 3 architectures that reach a 100% of precision. Architecture 1 with n = 10, architecture 2 with n = 9, and architecture 3 with n = 8. Therefore, architecture 3 is the best architecture valid (for cryptographic purposes), since it reaches a 100% of accuracy, with the smallest number of neurons.

It should be noted that an input set of 95 ASCII characters was used for this research, but this cryptographic method can work with more input characters. So, if we want that our network to encrypt/decrypt all 256 Extended-ASCII characters, we should only increase neurons in the intermediate layers.

Finally, this method of encryption can be a great change in current cryptographic methods. For future works, It would be interesting to analyze this method, in terms of efficiency and strength, and compare it with other common cryptographic methods such as: DES, AES, RSA, among others.

ACKNOWLEDGEMENTS

Authors thank to the SDAS Research Group (`www.sdas-group.com`) for its valuable support.

REFERENCES

Buduma, N. (2015). Fundamentals of deep learning: Designing next-generation artificial intelligence algorithms.

Charniya, N. N. (2013). Design of near-optimal classifier using multi-layer perceptron neural networks for intelligent sensors. *International Journal of Modeling and Optimization*.

Cheng, L. M. and Chan, C. (1998). Pseudo random generator based on clipped hopfield neural network. *in Proc. of the IEEE International Symposium on Circuit and system*.

Forouzan, B. A. (2007). Cryptography and network security.

Hadke, P. P. and Kale, S. G. (2016). Use of neural networks in cryptography: A review. *2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare*.

Jogdand, R. M. (2011). Design of an efficient neural key generation. *International Journal of Artificial Intelligence & Applications (IJAIA)*.

Karras, D. and Zorkadis, V. (2003). Improving pseudo random bit sequence generation and evaluation for secure internet communication using neural networks techquines. *International Joint Conf. on neural networks (IJCNN 2003)*.

Kinzel, W. and Kanter, I. (2016). Ido kanter.

Klein, E., Mislovaty, R., Kanter, I., Ruttor, A., and Kinzel, W. (2005). Synchronization of neural networks by mutual learning and its application to cryptography. *Advances in Neural Information Processing Systems*.

Lonkar, S. and Charniya, N. N. A. (2014). Neural networks based cryptography. *Vivekanaind Education Society's Institute of Technology*.

Matich, D. (2001). Redes neuronales: Conceptos basicos y aplicaciones.

Rojas, R. (1996). Neural networks: A systematic introduction.

Shi, Y. Q., Guorong Xuan, Zou, D., Jianjiong Gao, Chengyun Yang, Zhenping Zhang, Peiqi Chai, Chen, W., and Chen, C. (2005). Image steganalysis based on moments of characteristic functions using wavelet decomposition, prediction-error image, and neural network. *Institute of Electrical and Electronics Engineers(IEEE)*.

Stallings, W. (1998). Cryptography and network security: Principles and practice.

Tan, C. C. (1998). Autoencoder neural networks: A performance study based on image reconstruction, recognition and compression.

Volna, E., Kotyrba, M., Kocian, V., and Janosek, M. (2012). Cryptography based on neural network.

Wang, Z., Wang, N., and Shi, B. (2006). A novel blind watermarking scheme based on neural network in wavelet domain*. *6th World Congress on Intelligent Control and Automation,*.