ISSN 2278-3091
Volume 9, No.3, May - June 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse149932020.pdf
https://doi.org/10.30534/ijatcse/2020/149932020

Application of Machine Learning in Cryptography

Vikrant Shende[1], Meghana Kulkarni[2]
[1] KLS Gogte Institute of Technology, Belagavi, India, Vikrant_shende@git.edu
[2] VTU, Belagavi, India, meghanak@vtu.ac.in

ABSTRACT

Generally, neural networks have been extremely bad at cryptographic operations as they have a tough time carrying out an easy XOR computation. While that holds true, it ends up that neural networks can protect their private information from other neural networks by discovering unique structure of encryption and decryption, without being taught a specific algorithm. This is a slightly upgraded design used for the paper "Learning to Protect Communications with Adversarial Neural Cryptography". The model used in this paper is coded in python programming language, trained and run on raspberry-pi which provide dedicated hardware for the design.

Key words: Adversarial Neural Cryptography, Encryption, Decryption, Deep Learning, Neural Networks.

1. INTRODUCTION

The model used here is slightly simplified from the one described in the paper [1], the convolution layer width is decreased by half. In the original paper, there was a nonlinear layer after the fully-connected layer, that nonlinear has been gotten rid of here. These changes improve the effectiveness of training. The initializer for the convolution layers has switched to the tf.contrib.layers default of xavier_initializer instead of an easier truncated_normal. The system is coded in Python3 programming language using the Tensor-Flow framework. The raspberry-pi 4 is used as a dedicated hardware to train and run the model.

As shown in Figure-1. Sender takes a plain-text 'P' and key 'K' as input to produce a cipher-text 'C' at its output using encryption which is then sent out to Receiver, Receiver's goal is to decrypt this cipher-text 'C' using key 'K' to produce plain-text. Whereas Intruder's goal is to attempt to decrypt the cipher-text 'C' without having any other details to produce plaintext. The neural networks Sender and Receiver must discover their own encryption techniques to keep their communication private and protected without the Intruder neural network understanding anything about it. Sender and Receiver neural networks periodically enhance themselves by optimizing their own models to beat the best version of Intruder neural network.

Here none of the neural networks is given a specific cryptographic algorithm for encryption and decryption, using machine learning Sender and Receiver discover and optimize their own cryptographic algorithms over time to communicate with each other privately. Python and open-source library called Tensor-Flow is used to design and execute the cryptosystem. The cryptosystem training is carried out in two stages. In stage one Sender and Receiver neural networks are trained where they learn to encrypt and decrypt their messages for effective communication. In stage two Intruder neural network is trained, where it learns to intercept and decrypt their messages and output the original plain-text. Knowing that Intruder neural network can intercept and decrypt their messages, we train Sender and Receiver once again for them to enhance and optimize their system so that Intruder system can no longer decrypt their messages. Training Sender-Receiver and Intruder systems alternately, we are enhancing Sender and Receiver's communication over improving Intruder's ability to intercept and decrypt their messages. This forces Sender and Receiver system to continuously improve and develop new encryption technique to protect their personal communication.

We continue to train Sender and Receiver in addition to Intruder neural network which guarantees that Sender and Receiver keep improving their cryptographic strategies in order to beat the best variation of Intruder. Thus, supervised and unsupervised training elements are used in this cryptosystem to create a neural network architecture that is self-sufficient to learn mixing functions without implementing any particular encryption algorithm. Thus, to communicate securely, architecture of Sender, Receiver and Intruder which is a general neural network learns and evolves over time rather than using a specific algorithm [1].

First layer (FC) is a fully-connected layer with equal number of inputs and outputs. The plain-text and key is provided as input to this layer and output obtained from this layer is linear combination of all the input bits, which makes proper blending between secrete key and plaintext bits. This layer serves the purpose of permutation. This layer is followed by sequence of several convolution layers, the last layer produces

the cipher-text, which is equal in size to that of plain-text. The convolutions layers use some function on bits outputted by previous layers.

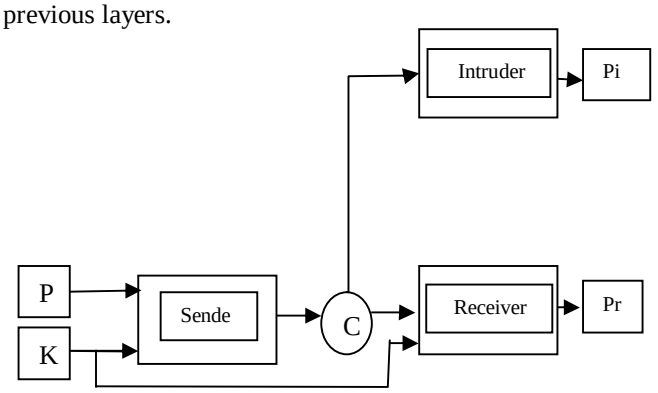

Figure. 1: Block diagram showing communication between Sender, Receiver and Intruder.

2. RELATED WORK

Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, Shimon Whiteson proposed, “deep distributed recurrent Q-networks” (DDRQN), which enable groups of agents to learn to fix communication-based coordination tasks. In order to successfully interact, agents in these jobs must first automatically develop and agree upon their own communication procedure. They provided empirical outcomes on two multi-agent knowing problems based on widely known riddles, showing that DDRQN can successfully fix such tasks and find sophisticated communication protocols. In addition, they provided experiments results that verify that each of the primary elements of the DDRQN architecture are important to its success [2].

Alexander Klimov, Anton Mityagin, and Adi Shamir, analyze the security of a brand-new key exchange suggested in "Secure exchange of information by synchronization of neural networks", which is based upon equally learning neural networks. This is a brand-new potential source for public key cryptographic designs which are not based upon number logical functions, as well as have small-time and memory details. In the initial part of the paper they review the strategy, describe why both parties get to a same key and why an opponent using a comparable neural network is not likely to reach the very same key. Nonetheless, in the 2nd part of the paper they show that this key exchange procedure can be barged in 3 different ways, and as a result it is completely insecure [3].

Neural Networks that can work with encrypted information is presented. This makes it possible for an data owner to send their data in an encrypted format to a cloud service that hosts the network. The encryption sees to it that the information stays personal since the cloud owner does not have access to the key needed to decrypt it. Nonetheless, cloud service can work with the encrypted information to make encrypted predictions and return these encrypted predictions to the cloud service user. The user can now decrypt these encrypted predictions to get back the original results. In the process cloud service provider cannot see the original information [4].

TensorFlow enables developers to try out unique optimizations and training algorithms. TensorFlow supports a variety of applications, with especially strong assistance for training and reasoning on deep neural networks. Several Google services utilize TensorFlow in production. It is launched as an open-source project and commonly utilized for artificial intelligence research study. They explain the TensorFlow dataflow model in contrast to existing systems and show the compelling efficiency that TensorFlow attains for several real-world applications [5].

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan,Pascal Germain,Hugo Larochelle,François Laviolette Mario Marchand, Victor Lempitsky, Presented a brand-new representation learning method for domain adaption, in which information at training and test time came from comparable however different sources. This method is directly inspired by the theory on domain adaption suggesting that, for efficient domain transfer to be accomplished, predictions should be made based upon functions that cannot discriminate between the training (source) and test (target) domains [6].

A technique which allows the discriminator network of a GAN to perform weakly localization of the items of interest. To attain this, they proposed a Teacher-Student training model as well as a unique kind of Soft-Class Activation maps. This scheme permits the discriminator to create weak annotation of the produced images which can be used for automatic annotation of produced images [7].

3. TRAINING/ RESULTS

We designed three networks namely Sender, Receiver and Intruder. These network takes N bit random plain-text and the key K and produce floating-point cipher-text. For N=16, 32 and 64 both plain-text and key are evenly distributed. Because of random key and plain-text generation there is a possibility of reusing. We executed our experiments in TensorFlow. We ran them on a raspberry pi.

In the beginning, Sender network combines two N-bit plain-text and key into 2N element vector using -1 and 1 to represent bits. The 2N X 2N FC layer processes this vector and passed through four 1D convolution layers. We used sigmoid function after every layer except last layer. The final layer reduces output to N elements, because of tanh unit. The tanh unit output ranges between [-1,1], which can be mapped to binary values. The Sender and Receiver networks are identical whereas the Intruder network only takes cipher-text thus has N X 2N FC layer.

We trained all the three neural networks in batch size of 4096 entries. The experimental results for batch size 4096 entries is presented. Adam optimizer is used with 0.0008 learning rate. Here learning rate is not minimize over time due to the fact that we desire Sender, Receiver and Intruder networks to respond to subtle alterations in other elements, until Sender and Receiver ideally attain a robust result to small modifications in Intruder network. Training switches between Sender/Receiver and Intruder, with Sender/Receiver training for one mini-batch and Intruder training for 2 mini-batches. We chose this proportion to give a minor computational advantage to the Intruder. We train Intruder network with the objective of reducing the squared error of its quote of 'C'.

To check learning of system in hiding data appropriately, we train a separate evaluator, which understands the distribution of 'C'. The separate evaluator tries to guess cipher-text relying just upon basic details, whereas the genuine Intruder network uses the intermediate output (D-public) and the cipher-text. If Intruder's reconstruction error equals separate evaluator error, we know that Intruder is not successfully drawing out details from the D-public and the cipher-text. Figure 2. reveals the results of adversarial training to discover to hidden 'C'. The orange "Intruder Reconstruction Error" line reveals the Intruder's reconstruction error. The blue shows "Receiver's reconstruction error", which drops over time up until Receiver is making the best possible forecast of 'D' offered the input. The public error ends up being low, however is a little higher than Sender's. As time elapses, Intruder's relative benefit becomes zero. Intruder is unable to reconstruct any more details about C.

Table 1: Sample values showing reconstruction error of Receiver and Intruder neural network

Iteration	Receiver's Reconstruction Error	Intruder's Reconstruction Error
0	7.98	7.99
200	7.91	7.52
400	7.88	7.56
800	5.33	6.11
1000	2.45	6.07
2000	0.83	6.76
4000	0.08	7.26
8000	0.02	7.46
10000	0.01	7.51
15000	0.01	7.58
20000	0.01	7.50
25000	0.01	7.58
26000	0.00	7.54
27000	0.01	7.62
27600	0.00	7.72

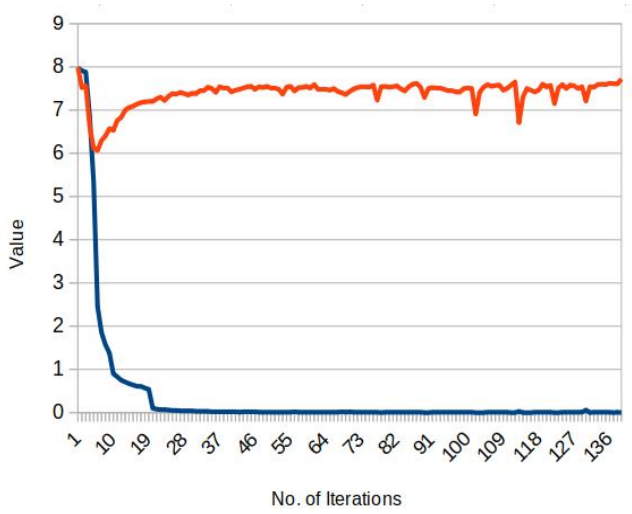

Figure 2: Reconstruction errors during training for Sender, Receiver and Intruder.

The results obtained from the experiments were quite impressive. As you can see in the Table-1, at 200 steps the reconstruction error is high for both Receiver and the Intruder, somewhere around 8000 training steps both Sender and Receiver start to rebuild the original message. Someplace around 10,000 training steps the Sender and Receiver networks seem to figure this out and Intruder's error rate climbs up once again. Simply put, Receiver has the ability to learn from Intruder's habits and safeguard the communication to prevent the attacker while still enhancing its performance.

Table 2: Sample values of Plain-text, Cipher-text and Hamming distance

Plain Text	Cipher Text	Hamming Distance
1111110011011100	1010010111100011	62.5
1010101010010010	1011010101001111	68.75
1100110010110011	1011000101100111	62.5
1000111111110111	1111000000011011	75.0
1000000100001011	1111010111110010	62.5
1111000000111010	1000111111010001	81.25

1110000100110010	1001111111001110	75.0
1011010011110101	1101100100001111	68.75
1111010101101001	1000001010001100	68.75
1001111000011101	1110010111100110	81.25
1101100100111100	1010011111000001	81.25
1111111011110111	1000000000001000	87.5
1110010011011011	1111111100110110	62.5
1111001101000100	1010100010101001	68.75
1010100110111101	1111111011001010	68.75
1111111111110101	1100001000001010	81.25
1110101010111001	1111010011010110	62.5
1110101011001111	1001000110010100	68.75
1000110110011110	1111101001111001	75.0
1101000101000110	1011111010111111	75.0

Table 3: Hamming Distance

AES	RSA	Machine Learning approach
87.5%	98.25%	65%

After the completion of training session, some random 16-bit binary numbers are given as input to the Sender neural network and its corresponding output is observed for calculating hamming distance. The sample 16-bit input, cipher-text output and hamming distance is shown in Table-2. A good encryption system will have more than 50% hamming distance. In our test results 10% of the inputs achieved a hamming distance of 70% and above. We achieved an average 65% hamming distance for our test results. Table-3 shows the average hamming distance calculated for AES, RSA and machine learning approach.

4. CONCLUSION

In this paper a neural network which continuously learns and optimizes its private communication is demonstrated. In the learning process none of standard cryptographic algorithms are used, rather it is based on a secrecy specification represented by the training goals. In this setup, attacker is modelled by neural network. Neural networks are not only helpful for cryptographic attacks but also can be used to protect data efficiently. While it looks unlikely that neural networks would become best at cryptanalysis, they might be rather effective in making sense of metadata and in traffic analysis. Although this is a relatively new research area, this topic has some appealing future scope. For instance, we might potentially utilize this innovation to make a few of existing cryptographic algorithms even stronger by adding a layer of artificial intelligence. Another possibility is to let neural networks discover their own encryption/decryption strategies which later could be evaluated.

REFERENCES

[1] Martín Abadi, David G. Andersen,"Learning To Protect Communications With Adversarial Neural Cryptography" . Google Brain

[2] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, Shimon Whiteson, "Learning to Communicate to Solve Riddles with Deep Distributed Recurrent Q-Networks". CoRR, abs/1602.02672, 2016a. URL http://arxiv.org/abs/1602.02672

[3] Alexander Klimov, Anton Mityagin, and Adi Shamir, "Analysis of Neural Cryptography". Y. Zheng (Ed.): ASIACRYPT 2002, LNCS 2501, pp. 288–298, 2002. Springer-Verlag Berlin Heidelberg 2002 https://doi.org/10.1007/3-540-36178-2_18

[4] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, John Wernsing, "CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy". Proceedings of the 33 rd International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

[5] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur,Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker,Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "TensorFlow: A system for large-scale machine learning". CoRR, abs/1605.08695, 2016b. URL http://arxiv.org/abs/1605.08695.

[6] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan,Pascal Germain,Hugo Larochelle,François Laviolette Mario Marchand, Victor Lempitsky, "Domain-Adversarial Training of Neural Networks". CoRR, abs/1505.07818, 2015. URL http://arxiv.org/abs/1505.07818.

[7] Dimitris Kastaniotis, Ioanna Ntinou, Dimitrios Tsourounis, George Economou and Spiros Fotopoulos, "Attention-Aware Generative Adversarial Networks (ATA-GANs)". 978-1-5386-0951-4/18/$31.00 #2018 IEEE.