DeepRing: Protecting Deep Neural Network with Blockchain

Akhil Goel*, Akshay Agarwal*, Mayank Vatsa*, Richa Singh*, and Nalini Ratha!
*IIIT Delhi and !IBM Research, NY, USA
*{akhil15126, akshaya, mayank, rsingh}@iiitd.ac.in, !ratha@us.ibm.com

Abstract

Several computer vision applications such as object detection and face recognition have started to completely rely on deep learning based architectures. These architectures, when paired with appropriate loss functions and optimizers, produce state-of-the-art results in a myriad of problems. On the other hand, with the advent of "blockchain", the cyber-security industry has developed a new sense of trust which was earlier missing from both the technical and commercial perspectives. Employment of cryptographic hash, as well as symmetric/asymmetric encryption and decryption algorithms, ensure security without any human intervention (i.e., centralized authority). In this research, we present the synergy between the best of both these worlds. We first propose a model which uses the learned parameters of a typical deep neural network and is secured from external adversaries by cryptography and blockchain technology. As the second contribution of the proposed research, a new parameter tampering attack is proposed to properly justify the role of blockchain in machine learning.

1. Introduction

The current era of artificial intelligence and machine learning is converting several dreams to reality. AI systems are getting implemented for making recommendations in social media and e-commerce sites to assisting medical professionals in medical diagnosis and robotic surgeries, and defending personnel with technologies such as drone surveillance. Such a wide spectrum usage of these technologies requires that the algorithms are secure.

A lot of this success can be attributed to deep learning architectures such as Convolutional Neural Networks (CNN) [13, 14]. CNNs contain blocks where each block can be referred to either as a convolutional layer or a combination of the convolutional, pooling, and non-linearity layers. The first layer which is an input layer passes the input samples to the first block of CNN, and this way information passes through the network to the last layer which makes the decision. For secure and correct use of these AI systems, fault-

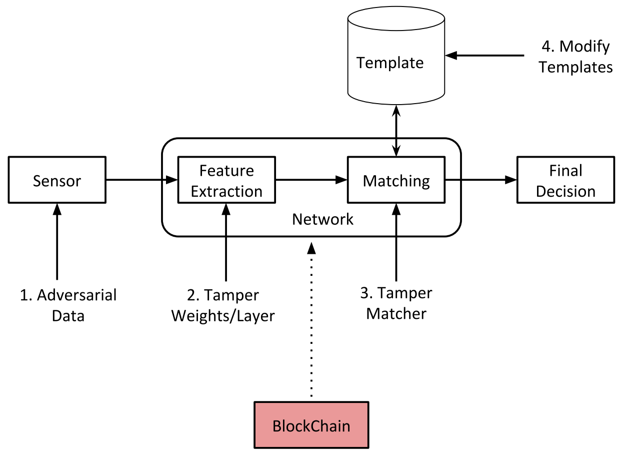

Figure 1. Vulnerabilities of artificial intelligence network and incorporation of blockchain for security.

less authentication of each block is a necessity. In other words, the accountability of each block which is missing in the original CNN models might be provided with the combination of the blockchain. Blockchain with its feature of data privacy, transparency, security, and authentication can help in the secure deployment of AI systems in the public domain. The data privacy in an AI system can be referred to as some information which is hidden from the general public which can be decrypted only using the private key of the authenticated owner of the system. On the other hand, the security aspect can be thought of as a guard who checks whether there has been any manipulation in the network architecture or not. The authentication feature can be referred to the property that the decision made by a particular block of the AI model would require the validation of other blocks connected with the block in concern.

Figure 1 shows the vulnerabilities of a typical artificial intelligence system. The attack on an AI system can be performed at an input level, architecture level, and decision level [17]. With the correct deployment of blockchain technology, attacks at the architecture and the decision levels can be avoided. For example, a recent algorithm,

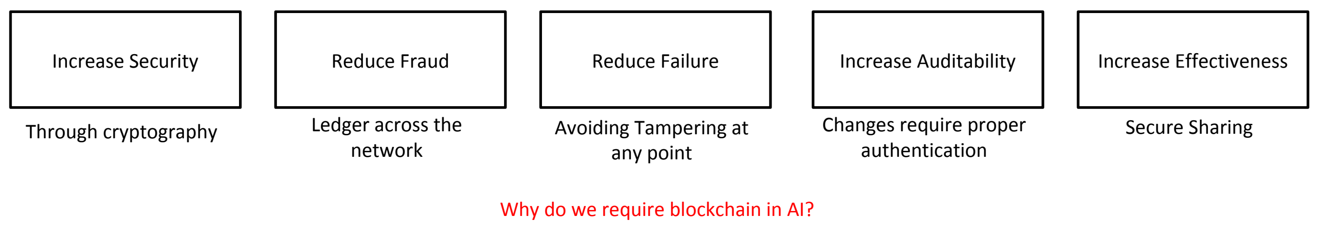
Increase Security
Through cryptography
Reduce Fraud
Ledger across the network
Reduce Failure
Avoiding Tampering at any point
Increase Auditability
Changes require proper authentication
Increase Effectiveness
Secure Sharing
Why do we require blockchain in AI?

Figure 2. Advantages of incorporation of blockchain in AI systems.

DeepChain, is proposed which is a collaborative training in a distributed, secure and decentralized environment [22]. It ensures the auditability of the training process and privacy of local gradients.

Blockchain is secure and powerful because of the following properties:

- Transitive Hash
- Cryptographic encryption at each step
- Decentralized nature

With the help of transitive hash and cryptography, the manipulation in any component of the network, i.e., features extraction and matching can no longer be performed. The alteration in any element will raise the alarm and inform the system that the operation implemented at a particular block is malicious, and the system can be restored at the previous checkpoint. Decentralization property, on the other hand, ensures that the entire control is not in the hands of a single entity. Just like how an unscrupulous chief executive officer of a company may present an inflated version of the company's assets to attract prospective shareholder's attention, an unethical model owner may use unfair means to boast about the model's performance. Decentralization makes sure that no foul play like this deceives the public. These properties mentioned above are required to deliver a safe and secure Deep Neural Network(DNN) model, and hence this makes blockchain an appropriate candidate for the job. Figure 2 summarizes the advantages of blockchain mechanism which can be provided to the AI system when successfully combined.

In this research, we have proposed the **'DeepRing'** architecture which combines CNN architecture with some features of the blockchain technology. Each block of DeepRing contains the following information which helps in authentication of the block against tampering:

- Hash of the current and previous block
- Public keys of the neighborhood layers
- Encryption keys of the current block
- Parameters of the current and the next layer

The DeepRing is able to detect any attack performed either at the parameter level or input level of each block (i.e., the network level attack). CNN models without blockchain have shown vulnerability against tampering. Whereas, as shown in Figure 1 the incorporation of blockchain in CNN (i.e., DeepRing) is successfully able to remove network level attack on CNN.

2. Components of the Proposed Solution

This section briefly explains the basic building blocks of both the technologies i.e., (i) blockchain and (ii) Asymmetric and Symmetric Key Cryptography.

2.1. Blockchain and Smart Contracts

Blockchain is a decentralized and a distributed ledger that records transactions between different parties. The recorded transactions are permanent and can be easily verified. Blockchain forms the basic building blocks of various crypto-currencies. Not directly relevant to the combination of CNN and blockchain, however, some recent work shows the potential of blockchain in various other fields such as smart energy and grids [2, 15], health-care [8, 16], and smart devices [20]. Recently, Delgado-Mohatar et al. [6] have presented their view about the combination of blockchain and biometrics to benefit both the technologies.

Smart Contracts [21] are pieces of codes that guarantee secure and credible transactions. One big advantage of smart contracts is that they eliminate the need for any third party altogether.

2.2. Asymmetric and Symmetric Key Cryptography

Asymmetric key cryptography [18] is a class of encryption algorithms that uses two keys instead of one for encryption and decryption purposes. One of these keys is the public key which is openly distributed and other is the private key which is kept exclusive. With asymmetric key cryptography, anyone can encrypt a message using the receiver's public key. This message can now only be decrypted using his private key. Asymmetric key cryptography enhances security but comes at a cost of reduced speed. Symmetric key cryptography uses only a single key to encrypt and decrypt data. These algorithms are generally faster than asymmetric

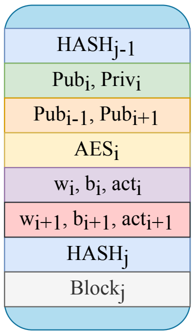

Figure 3. Depiction of layer i of a DNN represented as block j of the DeepRing

key algorithms but pose a challenge of securely transmitting the keys between parties.

3. DeepRing: Proposed Combination of CNN and BlockChain

In this section, an amalgamation of both important technologies i.e., CNN and blockchain is described through the proposed model referred to as DeepRing.

3.1. Notation

Here we define the notations used in Figure 3 and throughout the paper.

- $\mathbf{HASH_j}$: Hash of j^{th} block;
- $\mathbf{Pub_i}$: Public key of i^{th} layer;
- $\mathbf{Priv_i}$: Private key of i^{th} layer;
- $\mathbf{AES_i}$: AES key of i^{th} layer;
- $\mathbf{w_i}$, $\mathbf{b_i}$, $\mathbf{act_i}$: weight, bias, and activation function of the i^{th} layer;
- $\mathbf{w_{i+1}}$, $\mathbf{b_{i+1}}$, $\mathbf{act_{i+1}}$: weight, bias, and activation function of the layer next to the i^{th} layer;
- $\mathbf{Block_j}$: It is used to represent the block number for possible identification. It has no significance in current research.

3.2. Architecture

Figure 4 shows the transformation of a model from DNN architecture to a DeepRing architecture. The architecture of DeepRing is inspired by that of a blockchain. A blockchain is a linked list of ever-growing blocks with transaction records. DeepRing, on the other hand, is a closed chain of a finite number of blocks. Each block (except the ouroboros block) represents a layer of the deep neural network. Unlike a typical block in a blockchain, blocks of DeepRing serve the following purposes:

- Store the parameters of the layer
- Compute the output of the layer
- Update the ledger after output computation.
- Validate the output of the next layer

Figure 3 represents a block of the DeepRing. It comprises of the hashes of the current and previous block, public and private keys of the current layer, public keys of the layers appearing immediately before and after the current layer and AES key and model parameters of the current and the next layer. Just like blocks in a blockchain, blocks in DeepRing have a shared common ledger which stores the state of the model. The hash associated with a block is a function of the hash of the previous block and the parameters of the current and next layer parameters. Hash of a block j which corresponds to layer i in the DNN architecture is given by:

$$HASH_j = \Phi(HASH_{j-1}, params_i, params_{i+1}) \quad (1)$$

Here, Φ is any suitable hash function such as SHA256. After successful setup of the model in the DeepRing framework, hashes of all the blocks are stored by the ouroboros block. This is later used to track the compromised block in case of any tampering attack. One thing to note is that blocks in the ring **do not** follow any chronological order. They are arranged randomly. Even the blocks themselves are not aware of the sequential index of the layer whose purpose they serve.

3.3. Ouroboros Block

Ouroboros block is the start and end point of any DNN query. It is named after the greek symbol of a serpent eating its own tail called the ouroboros. It is the only known block of the ring. The weight matrix of the ouroboros block is the identity matrix, the bias is a zero vector and activation function is the identity function. Hence, output of an ouroboros block is the input itself. Ouroboros block stores the original value of the hash of the block preceding ($origHASH_{prev}$) it and has an extra parameter called as the authenticity parameter.

$$authenticity \leftarrow origHASH_{prev} == HASH_{prev}$$

Ouroboros block works in two modes:

- Query Mode ($authenticity$ = TRUE): This is the normal mode which takes in the input and sends back the output.
- Tracking Mode ($authenticity$ = FALSE): This mode is triggered when block parameters of any block in the network changes. Any change in any parameter is carried on to the hash of the last block. In this mode, the block suspends the general query business and tries to figure out the compromised block.

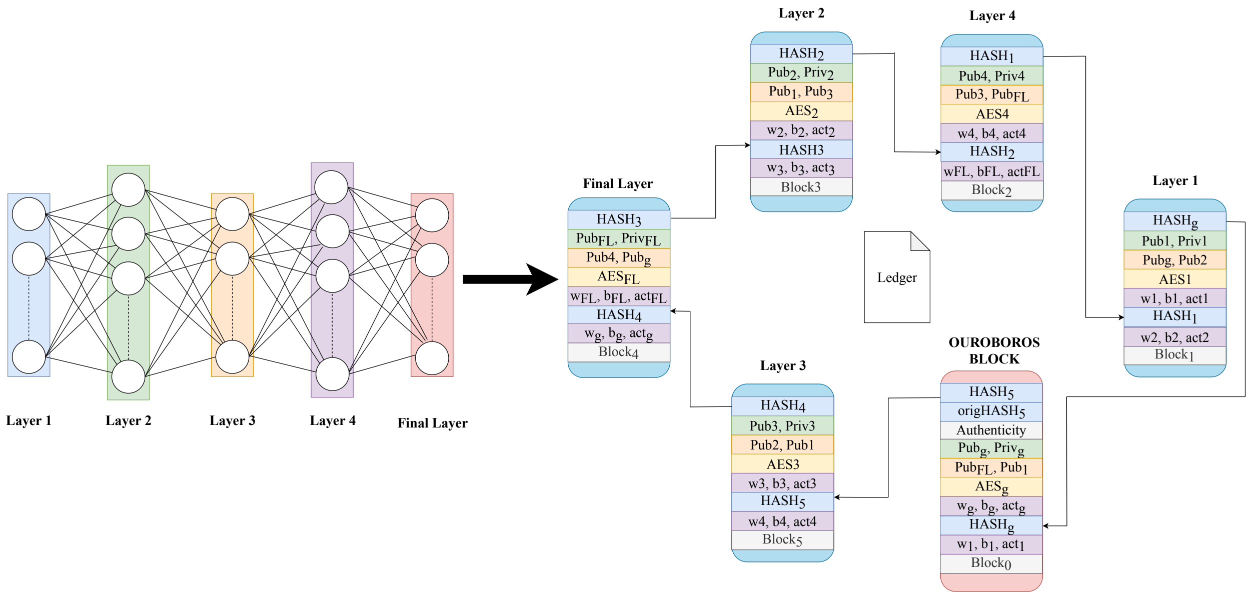

Figure 4. Transformation a typical DNN architecture to a DeepRing architecture

Hash of the ouroboros block is a function of only the layer parameters:

$$HASH_{ouroboros} = \Phi(params_{ouroboros}, params_1) \quad (2)$$

Since the ouroboros block has special responsibilities, this role should be taken by someone trusted, like the owner of the model.

3.4. Working of DeepRing

Working of the proposed framework is divided into various phases each of which is described below.

3.4.1 Query Phase

The querent encrypts the query with the public key of the only known block in the ring i.e., the ouroboros block. The encrypted query and the public key of the querent are updated on the ledger. DeepRing acknowledges a new query cycle with the ledger update by the querent. Updated ledger:

$$(encrypt_{Pub_O}(query), Pub_Q) \quad (3)$$

Here, Pub_O is the public key of the ouroboros block and Pub_Q is the public key of the querent.

3.4.2 Processing Phase

After processing the input to a block, each block updates the ledger with the following four items:

- Layer output encrypted by its AES key
- AES key encrypted by the public key of the next layer
- Signature of the layer
- Hash of the output of the next layer

After each ledger update, all the blocks check whether the update is signed by the layer that immediately precedes them. The signature here refers a message encrypted by the signer's private key.

For a layer i, let S_i be its signature, K_i be its encrypted AES key, X_i be its encrypted layer output and H_i be the hash of the next layer's output.

$$S_i = \Phi(encrypt_{Priv_i}(message)) \quad (4)$$

$$K_i = encrypt_{Pub_{i+1}}(AES_i) \quad (5)$$

$$O_i = \phi_i(w_i * x_i + b_i) \quad (6)$$

Here, ϕ_i is the activation function, x_i is the layer input and w_i and b_i are learned layer parameters of layer i.

$$X_i = encrypt_{AES_i}(O_i) \quad (7)$$

$$H_i = \Phi(\phi_{i+1}(w_{i+1} * O_i + b_{i+1})) \quad (8)$$

Here, ϕ_{i+1} is the activation function, and w_{i+1} and b_{i+1} are learned layer parameters of layer $i+1$. Suppose this layer i updates the ledger with S_i, K_i, X_i and H_i, then at any layer j:

$$assert(\Phi(message) == decrypt_{Pub_{j-1}}(S_i)) \quad (9)$$

This assertion is true only when $j-1$ equals i. In other words, the assertion is true for the $(i+1)^{th}$ layer. After signature verification, K_i is decrypted to retrieve the AES key of the previous layer.

$$key = decrypt_{Pvt_{i+1}}(K_i) \tag{10}$$

This key is the required key to decrypt the input to $(i+1)^{th}$ layer.

$$x_{i+1} = decrypt_{key}(X_i) \tag{11}$$

Layer $i+1$ updates the ledger with S_{i+1}, K_{i+1}, X_{i+1} and H_{i+1} and the cycle goes on till the control falls back to the ouroboros block. This two-step authentication guarantees the following security concerns:

- Signature verification ensures that a layer listens to only authorized senders;
- AES key decryption ensures that only the intended layer gets to see the layer output.

3.4.3 Delivery Phase

The processing cycle completes when the control falls back to the ouroboros block. When the block verifies that the last ledger update has been made by the final layer of the model, it follows the above procedure to retrieve the model output ($x\_ouroboros$). This output is encrypted using the public key of the querent (Pub_Q). This ensures that no one but the querent can access the model output results. After successful completion of a query, the ordering of the blocks along with the AES keys of the layers is changed to maintain randomness in the network.

$$model\_output = encrypt_{Pub_Q}(x\_ouroboros) \tag{12}$$

3.4.4 Tracking Phase

This phase is triggered when a change in any model parameters leads to a change in the hash of the containing block and subsequently a change in the hash of the last block. In this phase, the ouroboros block tries to find the first block whose hash does not match with the hash in the records and forces it to restore back to the state in which it previously (at the time of framework setup) matched. This process is repeated until the hash of the last block is restored to the original value and *authenticity* signal again turns back on.

3.5. Validation and Consensus

Validation Principle: We define the output validation principle for any layer i as follows:

$$assert(H_{i-1} == \Phi(\phi_i(w_i * x_i + b_i))) \tag{13}$$

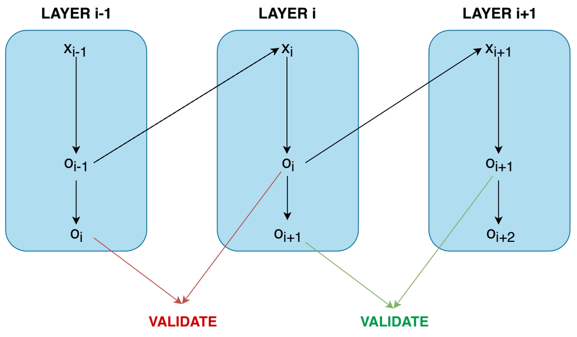

Figure 5. Validation of layer outputs. Here x_i and o_i refer to input and output of layer i respectively.

Proposition 1. *Layer i is compromised if and only if the validation principle for layer i is violated.*

Proof. First, we prove the necessary condition ($\rightarrow$)

Let us assume that layer i is compromised. Any layer is considered to be compromised if the input that it operates on is actually perturbed. If x_i is perturbed then from Figure 5 we have:

$$o_{i-1} \neq x_i$$

$$o_i|_{layer\,i-1} \neq o_i|_{layer\,i}$$

Hence by the property of hash algorithms that hash of non-equal elements is unequal, we conclude that the validation principle does not hold.

We now present the sufficient condition ($\leftarrow$). Let layer i be the first encountered layer (after the ouroboros block) for which the validation principle does not hold. Hence we have,

$$H_{i-1} \neq \Phi(\phi_i(w_i * x_i + b_i))$$

$$o_i|_{layer\,i-1} \neq o_i|_{layer\,i}$$

Therefore, either $o_i|_{layer\,i-1}$ is not genuine which implies that layer $i-1$ is compromised or $o_i|_{layer\,i}$ is not genuine which implies layer i is compromised.

$o_i|_{layer\,i-1}$ has to be genuine because layer $i-1$ being compromised implies that the validation principle failed for $(i-1)^{th}$ layer which contradicts the initial assumption that i is the first layer for which the validation principle does not hold. Hence proved that layer i is compromised.

This concludes the proof of Proposition 1. $\square$

Apart from producing the layer output, blocks in DeepRing perform another major task and that is to validate the output of the block that follows. Any layer i updates the ledger with S_i, K_i, X_i and H_i. After extracting x_{i+1} from X_i as explained above, layer $i+1$ performs the assertion of the validation principle. By Proposition 1, we can conclude that consensus on successful computation of layer i output is achieved only when this assertion holds true.

4. Adversarial Attack and Security Analysis

This research aims to defend the DNN models against adversaries who either aims to perturb the learned parameters of the model or leverage the model information to craft adversarial samples in a white-box framework[1]. The section defines the proposed attack which tries to manipulate the network parameter for possible malfunctioning of the network. The section further presents security analysis of the proposed DeepRing architecture with respect to two thread models, i.e. perturbing the layer parameters and perturbing the input of the block.

4.1. Adversarial Parameter Tampering

As shown in Figure 1, an adversary can attack the deep learning model at multiple levels such as by attacking the input or tampering the network parameters. While a lot of work has happened in attacking at the input level [1, 3, 7, 9, 10], very limited research has focused on adversarial attack on network parameters. In this section, we present a parameter altering (attack) algorithm which perturbs parameters with the largest consequences to the network. The proposed attack in a true sense justifies the role of blockchain in deep CNN models and shows it's effectiveness in curbing the attack on network level (as shown in Figure 1).

Recently, Keshari et al. [11] propose to reduce the number of parameters of CNNs by introducing a strength parameter for filter weights. We follow a similar approach and use the strength of weights/kernel matrix as a measure to decide whether it should be perturbed or not. Since the goal is to achieve maximum deviation with a minimum level of distortion, we only perturb the parameters which are the most important. Keeping the trained parameters of the network fixed, we associate each layer with a logistic importance parameter p, and train the network again to compute the importance of each layer weights. Larger the value of p, more important are the weights of the layer in reaching the final decision. We perturb the weights of the most important layer (largest p value) with Gaussian noise and monitor the network performance. Naturally, since the parameters are perturbed, the performance of the model is negatively affected. The level of deviation depends on the quantity of noise added.

Algorithm 1 presents the pseudo-code of the proposed attack. $\odot$ operation refers to the Hadamard product between the weights matrix and strength parameter, $*$ operation refers to convolution operation in the context of convolutional layers and matrix multiplication operation in the context of dense layers. Method $Noise(\mu, \sigma, shape)$ returns a Gaussian matrix of specified shape with mean μ and standard deviation σ and method $sigmoid$ refers to the logistic Sigmoid function.

[1]white-box framework is defined as a framework where the attacker has complete access to the network information such as parameters and gradient.

Algorithm 1 Proposed Parameter Tampering Attack

```
1:  procedure ATTACK($model, img, labels, \mu, \sigma$)
2:      $i \leftarrow 0$
3:      $n \leftarrow num(model.layers)$
4:      while $i < n$ do
5:          $parameters_i.trainable = False$
6:          $W_i, b_i = parameters_i$
7:          $initialize\ p_i$
8:          $output_i \leftarrow \phi((W_i \odot sigmoid(p_i)) * input_i + b_i)$
9:          $i \leftarrow i + 1$
10:     end while
11:     $model.train(img, labels)$
12:     $P = p_j\ s.t.\ \{\forall i \in n \mid \exists j \in n : p_j \geq p_i\}$
13:     $noise\ =\ Noise(\mu, \sigma, weights_j.shape)$
14:     $weights_j \leftarrow weights_j\ +\ noise$
15: end procedure
```

4.2. Security Analysis

In this section, we analyze the robustness of the proposed DeepRing architecture in various threat model scenarios. An adversary or a compromised block can conspire against the network in following two ways (and their security analysis thereof):

- **Perturb the layer parameters of the block:** Hash of any block is a function of the parameters of the layer it represents. Any change in the values of these parameters would correspond a change in the hash of the block. Since the recomputed hash is a parameter for the hash of the next block, the hash of the next block changes too. This goes on until the hash of the last block is changed. This change triggers the authenticity signal of the ouroboros block to turn off and activates its tracking mode. Normal query phase is resumed once the parameters of the compromised blocks are restored.

- **Perturb the input to the block:** Changing the input of a layer does not affect the hash of that layer. However, it affects the validation assertion explained previously. Consider a block j representing layer i of the DNN model. Assuming it is under the influence of an adversary and is compromised, it tries to perturb or change its input to produce the wrong output. Proposition 1 explains that in such a case, validation fails and hence consensus can not be achieved. Since, a violation of validation principle at any layer i indicates that it has been compromised, in case of an actual violation, the output from the previous layer is sent back to the current layer.

Table 1. Number of layers (n) vs probability of guessing the correct sequence of layers.

n	Probability (%)
2	50
3	16.66
5	0.83
7	0.019
10	2.7x10^{-5}

Table 2. Object recognition performance of a VGG-19 [19] before (i.e., original) and after the parameter tampering and input perturbation attacks.

Type	Model	MNIST	CIFAR-10	Tiny-ImageNet
Original	DNN	**99.07**	**83.89**	**76.12**
Parameter Tampering	DNN	51.20	63.18	41.96
	DeepRing	**99.07**	**83.89**	**76.12**
Input Perturbation	DNN	81.54	74.88	58.21
	DeepRing	**99.07**	**83.89**	**76.12**

Table 3. Object recognition performance of the described neural network (NN) before (i.e., original) and after the parameter tampering and input perturbation attacks.

Type	Model	MNIST	CIFAR-10
Original	NN	**98.10**	**62.35**
Parameter Tampering	NN	64.29	50.11
	DeepRing	**98.10**	**62.35**
Input Perturbation	NN	77.89	44.37
	DeepRing	**98.10**	**62.35**

White-box attack strategies such as C&W L_2 [4] and EAD [5] depend on model properties such as gradients to craft adversarial samples. Due to complete random and encrypted nature of DeepRing, the simplest properties of the model such as the route of information flow between blocks is hidden from the external world. Anticipating the next block which proceeds with the computation will entirely be based on a random guess. The probability to guess the correct sequence of events decreases dramatically with an increase in the number of layers. For the DeepRing architecture:

$$Prob(guess\,the\,correct\,sequence) = \frac{1}{n!} \qquad (14)$$

Table 1 shows that the probability of guessing the correct sequence order drastically decreases with an increase in the number of model layers. Therefore, it is difficult to attack the DeepRing network by leveraging gradient flow in the model as proposed in most of the existing adversarial attack algorithms.

5. Experimental Results

In this section, we describe the experiments conducted to evaluate the validity and robustness of the proposed approach. We modeled VGG-19 architecture [19] trained on MNIST[2], CIFAR-10 [12], and Tiny-ImageNet[3] in a DeepRing framework. We also considered a neural network with five dense layers with the following properties:

- Number of nodes in each layer [900, 600, 300 , 100, number_of_classes]
- ReLU activation for all layers except the output layer. Softmax activation is used for the output layer.

We trained this network on MNIST and CIFAR-10 datasets. Further, we experimented with the following two cases:

- **Attacking the model using the tampering attack:** We applied the parameter tampering attack on DNN architecture and the DeepRing architecture. For the case of DeepRing, tampering the parameters alerts the ouroboros block and activates its tracking mode. The parameters of the affected blocks are restored from the last stable checkpoint.
- **Compromising a block by changing its input:** In order to monitor the behavior of the DeepRing architecture on input perturbations, we perturb the third convolutional layer input of the VGG19 model and the third dense layer input of the neural network architecture described above. Perturbing the inputs leads to the failure of the validation clause which indicates compromise.

Table 2 shows the performance of VGG-19 with and without blockchain for the object recognition task on MNIST, CIFAR-10, and Tiny ImageNet databases. Table 3 shows the performance of the defined neural network with and without blockchain for the object recognition task on MNIST and CIFAR-10. The networks are attacked using the methods listed above. When the original model is used for recognition, the VGG 19 model yields 99.07%, 83.89%, and 76.12% object recognition accuracy on MNIST, CIFAR-10, and Tiny ImageNet databases, respectively. However, because of no security mechanism deployed in the individual block of a typical model, it shows vulnerability towards attacks and performance suffers a significant drop. The performance on CIFAR-10 drops by 20.71%, whereas, on MNIST and Tiny ImageNet datasets, drop of more than 47% and 34%, respectively is observed. However, as described in section 3, the proposed DeepRing model is fault free because of multiple authentication blocks such as validation/consensus and Hash functions. Therefore, we do not observe any reduction in performance for the proposed DeepRing. In case of compromising a block by changing its input, DeepRing model will also trigger a

[2]http://yann.lecun.com/exdb/mnist/

[3]https://tiny-imagenet.herokuapp.com/

violation and therefore, it can inherently provide adversarial attack detection mechanism. In our experiments, we observe that the proposed DeepRing yields 100% accuracy for detecting perturbations of input to a block.

6. Conclusion

In this research, we have shown that the synergy between the power of DNN models and security of blockchain can create tamper-proof models. The characteristic properties of blockchain such as security and accountability are provided to the CNN models by employing cryptographic techniques and by decentralizing layer operations. The blocks of CNN models are placed in a random order where the neighborhood blocks contain the information of the next legitimate block. Hiding the network architecture and parameters from an adversary prevents the threat of any white-box adversarial attack. Tampering in any block changes the hash of the current and the subsequent blocks thereby highlighting the performed attack. In this way, the transparency between the blocks and the entire network is increased. However, this enhanced version of security comes at a price of increased computational complexity of performing expensive cryptographic functions and protocols. In this research, through experiments, we have shown that the network is protected against adversarial tampering and layer perturbation attacks. In the future, we will extend the approach to make it efficient in terms of computational complexity and defend models against input image perturbation.

References

[1] A. Agarwal, R. Singh, M. Vatsa, and N. Ratha. Are imageagnostic universal adversarial perturbations for face recognition difficult to detect. *IEEE International Conference on Biometrics: Theory, Applications and Systems*, 2018. 6

[2] S. Aggarwal, R. Chaudhary, G. S Aujla, A. Jindal, A. Dua, and N. Kumar. Energychain: Enabling energy trading for smart homes using blockchains in smart grid ecosystem. In *ACM MobiHoc Workshop on Networking and Cybersecurity for Smart Cities*, page 1, 2018. 2

[3] N. Akhtar and A. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018. 6

[4] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, pages 39–57, 2017. 7

[5] P. Chen, Y. Sharma, H. Zhang, J. Yi, and C. Hsieh. Ead: elastic-net attacks to deep neural networks via adversarial examples. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 7

[6] O. Delgado-Mohatar, J. Fierrez, R. Tolosana, and R. Vera-Rodriguez. Blockchain and biometrics: A first look into opportunities and challenges. *arXiv preprint arXiv:1903.05496*, 2019. 2

[7] A. Goel, A. Singh, A. Agarwal, M. Vatsa, and R. Singh. Smartbox: Benchmarking adversarial detection and mitigation algorithms for face recognition. *IEEE International Conference on Biometrics: Theory, Applications and Systems*, 2018. 6

[8] W. J Gordon and C. Catalini. Blockchain technology for healthcare: facilitating the transition to patient-driven interoperability. *Computational and structural biotechnology journal*, 16:224–230, 2018. 2

[9] G. Goswami, A. Agarwal, N. Ratha, R. Singh, and M. Vatsa. Detecting and mitigating adversarial perturbations for robust face recognition. *International Journal of Computer Vision*, 2019. doi: 10.1007/s11263-019-01160-w. 6

[10] G. Goswami, N. Ratha, A. Agarwal, R. Singh, and M. Vatsa. Unravelling robustness of deep learning based face recognition against adversarial attacks. *Association for the Advancement of Artificial Intelligence*, pages 6829–6836, 2018. 6

[11] R. Keshari, M. Vatsa, R. Singh, and A. Noore. Learning structure and strength of cnn filters for small sample size training. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9349–9358, 2018. 6

[12] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 7

[13] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015. 1

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[15] A. Magnani, L. Calderoni, and P. Palmieri. Feather forking as a positive force: incentivising green energy production in a blockchain-based smart grid. In *ACM Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, pages 99–104, 2018. 2

[16] M Mettler. Blockchain technology in healthcare: The revolution starts here. In *IEEE International Conference on e-Health Networking, Applications and Services*, pages 1–3, 2016. 2

[17] N. K. Ratha, J. H. Connell, and R. M. Bolle. Enhancing security and privacy in biometrics-based authentication systems. *IBM systems Journal*, 40(3):614–634, 2001. 1

[18] G. J Simmons. Symmetric and asymmetric encryption. *ACM Computing Surveys*, 11(4):305–330, 1979. 2

[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 7

[20] Q. Stokkink and J. Pouwelse. Deployment of a blockchain-based self-sovereign identity. *arXiv preprint arXiv:1806.01926*, 2018. 2

[21] N. Szabo. Smart contracts: building blocks for digital markets. *EXTROPY: The Journal of Transhumanist Thought,(16)*, 18, 1996. 2

[22] J Weng, Jian Weng, J Zhang, M Li, Y Zhang, and W Luo. Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive. *Cryptology ePrint Archive, Report 2018/679*, 2018. 2