Enhanced Deep Learning Anti-Data Corruption

Jose Jimenez Espada (6140508)
Knight Foundation School of Computing and Information Sciences (KFSCIS)
Florida International University

***Abstract*—Data corruption is a major issue for personal computers, databases, and commercial computers. Although there are current methods to detect data corruption and restore data corruption, advances in algorithms have led us to integrate AI algorithms to restore corrupted data. Deep learning is advanced enough to help reverse any damages done by corruption. This paper will show how a combination of deep learning and double layered checksums to prevent and reverse data corruption.**

***Index Terms*—deep learning, checksum, heuristic, gradient descent, data corruption**

I. Introduction

Data corruption is a major issue in the computing industry. Users face corruption for several reasons. Data corruption is commonly caused by hardware failures and interruption. Most personal computer users do not understand the risks associated with interrupting tasks while the OS is storing data in memory because this causes corruption. Integration of AI at the OS system level helps prevent this common silent issue, which most are not even aware of. The operating system on most personal computers states that the corrupted data is - most of the time - damaged beyond repair. This paper wants to change this with this new method.

Beyond personal computers, commercial computers and databases suffer from corruption, not because of recklessness or lack of knowledge, but because attackers corrupt data as a form of cyberattack [1]. At this level, it is crucial for these industries to have such operating algorithms to help reverse what was labeled irreversible.

AI integrated in operating systems is underutilized, and it aids with the obstacles and hurdles that operating systems go through. It should be a necessary tool to aid loss of important or useful data.

II. Method

A. Limited Method

Since the resources used are limited for re-implementing a universal algorithm that will help restore corrupted data, for all if not most data files. For the sake of the example, the project will be based on a small sample, and although the sample is small, it is practical and can be expanded at a larger scale. The vision of the project is to make use of its application to the fullest extent. If it succeeds with image corruption, in theory, it should restore any type of corrupted file. Only sections of the file will be restored.

B. Applied Method

This project used a sample of corrupted images, and will use *Deep Learning Neural Networks* to restore corrupted images. Hypothetically speaking, this algorithm alone is powerful enough to restore corrupted data, but it may still fall short. The project will use double layered checksums to prevent corruption go undetected [2]. Even if corruption were to occur, multiple layers of checksums, would make it easier for the Neural Network to converge to an uncorrupted file. The method will have two parts: *Deep Learning Neural Network* and *multiple layer checksums*.

C. Deep Learning Neural Network

The neural network will consists of two layers and will use the backpropagation algorithm to converge to the correct uncorrupted data [5]. In this limited and small scale, the project will concern about which part of the binary data file is corrupted and how it is likely to be restored. With the limited resources used for testing and implementation, the inputs will consist of the first line of characters in the data file. The checksums will detect if there is any corruption on the file, and then it will try to salvage any part of the file to restoration. In the equation (1) we can see how the processing elements of the second layer is calculated. Equation (2) consists of the general weighted calculation of the neural network. Equation (3) consists of the activated function "SoftMax" which is easy to implement and derive from [5].

$$L_{1_p} = a(f(x_1, x_2, ..., x_n)) \tag{1}$$

$$f(x_1, x_2, ..., x_n) = \sum_{i=1}^{n} w_{p_i} * x_i \tag{2}$$

$$a(x) = max(0, x) \tag{3}$$

The second layer will consists of four processing elements which will help find the more complex decision boundary. It will replicate the calculations of the first layer with the small of adjustment of weight sums and parameter input sizes.

In the project's neural network, the parameter input size will be the file. For the experiment sake, we analyzed the first line of the binary version of the image which is 42 input elements. These calculation will go to a hidden layer, for the experiment sake, we have chosen 20 processing elements in the first hidden layer, because it is one of the most effective size. The output layer will be the same size of the input layer, since the algorithm generally rewrites the binary file to what it believes it is supposed to be anyway. In figure (1), we can see

the modeled neural network on a small scale for explanation purposes. For demonstration purposes, figure (1) is only shown on a small scale, because the actual diagram of the neural network used is large enough to not be as comprehensive.

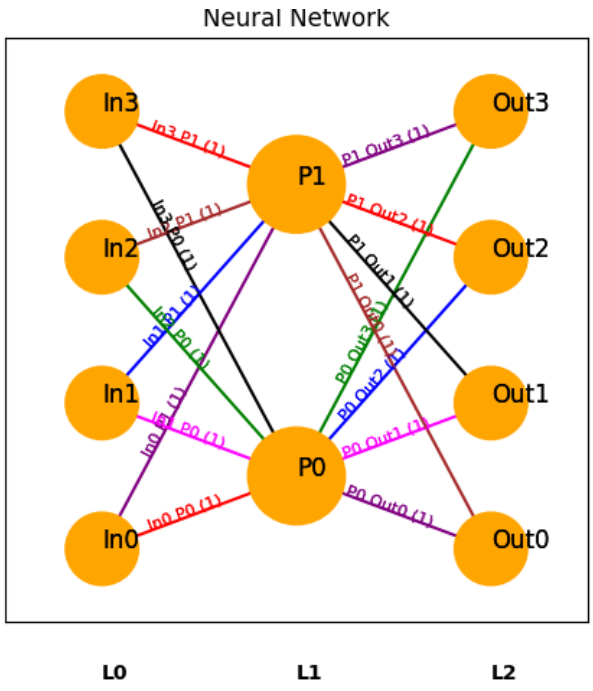

Fig. 1. An example of the neural network used for regenerating the binary file on a small scale.

D. Double Layered Checksums

This method makes the process less error prone and makes the process more efficient, because it will be the neural network's heuristic. Because neural networks are not perfect, no matter how much you train it, there must be alternative measures to prevent from over reliance of neural networks. This also prevents the hallucination problem of neural networks from occurring [7]. Instead of one checksums validating of whether or not the file is corrupted, multiple layered checksums make the process more difficult to corrupt. The assignment of checksums will be based on sections of the data file and will be equally divided. For the sake of our limitations, we will base this on each binary line on file, and assign their checksums throughout the file.

$$C_i^1 = f(x_i, x_{i+1}, ..., x_n) \tag{4}$$

Equation (4) represent the first layer of checksums. The function is the calculation and procedure for converting a stream of data elements into a checksum to verify if the file is corrupted.

$$C_i^2 = f(C_i^1, C_{i+1}^1, ..., C_n^1) \tag{5}$$

Equation (5) represents the second layer of checksums, which is a recursion of the first layer. The checksums of the data streams are being verified and treated as if they are data elements, being verified across the data stream.

E. Proposed Format

In order for this to work, we must restructure how data is saved. Double layered checksums require us to save metadata of both each line in the binary data file, and checksums of every line on file. Although the file will be substantially larger, the data file is less likely to be corrupted, if this is implemented as the main storage format for the OS.

F. Ideal Method

On a larger scale, there will be a larger deep learning neural network with more layers, since every file-type has a different logical structure, thus requiring more complexity. Checksums verification method would have more layers, ensuring full security of the file, and further detection aid for the file. The training would be more vast and more intensive and adds in validation and testing.

III. Experiment and Results

A. Equipment and Resources

The project will be operating on a laptop, with an Intel Quad Core I-7 with 1.9 GHz, 24 GB of RAM, and it operates Windows 10. The selected programming language of the project is Python, and the project will use it own version of neural network, in other words, it will be a hard coded neural network.

B. Dataset

Our dataset will be based on a series of pictures of a house. We will only reverse corruption on images of houses, since it is sufficiently large enough to have a reliable model. The dataset will consist of 14 images, and a series of correct checksums. Since it is risky downloading corrupted files from the internet or other sources, the project contains an algorithm that replicates the image from the binary file and modifies it (intentionally corrupting it) in order to train the model. The model create 100 different corrupted versions of a single image, but all of this will be fed into the model for training, and ideally it will uncorrupt these series of files.

Fig. 2. An image of the sample house used for training the model.

C. Gradient Descent

The deep learning needs to adjust the weights and bias consistently, yet logically. Using the gradient descent approximation theorem, a randomized genetic algorithm for gradient descent, we will find the ideal weights efficiently and logically [5]. This method is used due to it may finding the results quicker than expected, and it also prevents the local minima and maxima problem being labeled as global minima or maxima [6]. The approximation theorem that the project used is attempting with two layers of random weights and biases and adjusting from there, based on the steepest descent from the initialization. The program will pick the best model, and make it have a child with slight modifications in hope of improvement.

D. Training and Loss

In the training, it shows how much potential this concept has in restoring corrupted data. The training alone, has brought significant results, although it is not up to industry standards, for the amount of training the model has seen makes it reliable. As we can see in figure (2), we have approximated our model

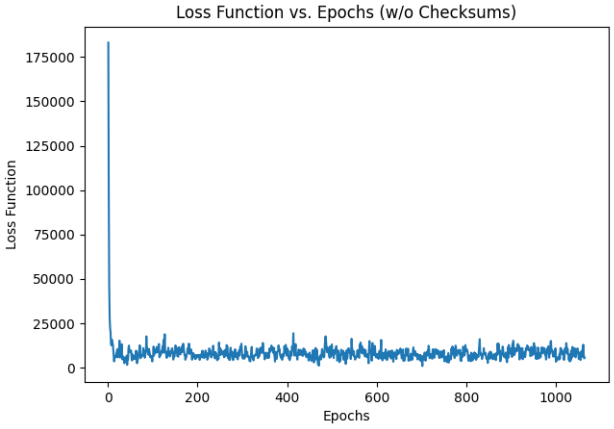

Fig. 3. The training loss as the model was being trained to repair slightly corrupted images into non-corrupted images.

to converge the reversal of the neural network. The training set is short enough, yet effective. Although the model's loss function is not quite zero, this is acceptable enough to restore the corrupted datafile effectively.

IV. Related Works

There is many methods for anti-data corruption, and most if not all of them, do not use the method of AI to solve the problem. The papers and resources reviewed talks about methods on preventing data corruption to happen in the first place. Most research papers analyzed are based on detecting data corruption, not reversing them.

In another paper, it talks about a useful framework named Proactive Checking Framework (PCF) in how this framework automatically deals with database corruption on a state basis [3].

There is a paper that uses multiple layered checksums [1]. In the experiment, we took inspiration and recreated it in a small scale, by instead of having in multiple layers, we have it in double layers.

Most of the papers analyzed speak about how to prevent data corruption from occuring in the first place. For example, even-bit parity and odd-bit parity can both prevent corruption and reverse corruption.

V. Conclusion

This experiment is valuable and should be further enhanced and analyzed. Unfortunately, because of lack of better equipment, testing this had to be both specific and practical at the same time. It is found that Deep Learning does help restore data files, if trained and tested effectively, and thoroughly. Double layered checksums does reduce processing time, because it will detect even the least corrupted file to be fixed, due to a series of sophisticated signatures. Although the paper was limited in scope and training, images were only used and not actual documents, it was effective and functional and highlighted a revolutionary concept, which is something most OS do not have.

References

[1] D. Barbara, R. Goel, and S. Jajodia, "Using Checksums to Detect Data Corruption."

[2] P. Bohannon, R. Rastogi, S. Seshadri, A. Silberschatz, and S. Sudarshan, "Detection and recovery techniques for database corruption," IEEE Transactions on Knowledge and Data Engineering, vol. 15, no. 5, pp. 1120–1136, Sep. 2003, doi: https://doi.org/10.1109/tkde.2003.1232268.

[3] N. Borisov, S. Babu, N. Mandagere and S. Uttamchandani, "Dealing proactively with data corruption: Challenges and opportunities," 2011 IEEE 27th International Conference on Data Engineering Workshops, Hannover, Germany, 2011, pp. 34-39, doi: 10.1109/ICDEW.2011.5767656.

[4] Mark Woodard, Sahra Sedigh Sarvestani, Ali R. Hurson, Chapter Two - A Survey of Research on Data Corruption in Cyber–Physical Critical Infrastructure Systems, Editor(s): Ali R. Hurson, Advances in Computers, Elsevier, Volume 98, 2015, Pages 59-87, ISSN 0065-2458, ISBN 9780128021323, https://doi.org/10.1016/bs.adcom.2015.03.002. (https://www.sciencedirect.com/science/article/pii/S0065245815000285)

[5] K. V. Sree Bai and M. Thirumaran, "Survey of Deep Learning Techniques for Malware Detection: Insights, Challenges, and Future Directions," 2024 4th International Conference on Soft Computing for Security Applications (ICSCSA), Salem, India, 2024, pp. 320-324, doi: 10.1109/ICSCSA64454.2024.00057.

[6] A. Rakotomamonjy, S. Koço and L. Ralaivola, "More efficient sparsity-inducing algorithms using inexact gradient," 2015 23rd European Signal Processing Conference (EUSIPCO), Nice, France, 2015, pp. 709-713, doi: 10.1109/EUSIPCO.2015.7362475.

[7] Elfman, Liz. "What Are AI Hallucinations? Examples Mitigation Techniques." Data.world, 10 Sept. 2024, data.world/blog/ai-hallucination/. Accessed 28 Mar. 2025.