Repairing Faulty Neural Networks

1st Yih Cheng
Dept. of ECE
University of Toronto
Toronto, Canada
joeycy.cheng@mail.utoronto.ca

***Abstract*—This study looks to investigate the damaging and repairing of neural networks through conducting experiments on models trained with MNIST dataset [1]. We first demonstrate the baseline relationship of damaged networks and accuracy. Two repairing mechanics, one being neuron replication and the other being modifying the training process, will be introduced and evaluated. Analyzing these results and methods can provide us a deeper understanding of how the traditional neural networks react to damages and how they provide fault tolerance, whether implicitly or explicitly.**

***Index Terms*—neural network, fault tolerance**

I. Introduction

Neural networks have gained widespread recognition thanks to its remarkable performance across various tasks and applications in recent years. However, most research focuses on enhancing performance, while few are targeted towards fault tolerance in these models. Some notable works that address this issue are [2], which introduced augmentation techniques and proves to enhance the robustness of the model. [3] demonstrates that fault tolerance can be achieved through modifying learning algorithms. Additionally, [4] offers a deep review of current studies regarding fault tolerance in neural networks.

This study primarily investigates strategies for enhancing fault tolerance in faulty neural networks trained on MNIST dataset [1]. Experiments are conducted on three types of neuron damage: stuck-at minimum, stuck-at maximum, and random noise faults. Two different repairing techniques are considered. The first approach replicates neurons on trained models in view of providing a higher fault tolerance. The second approach tries to modify the training process by introducing random faults into the network during training. In addition, the comparisons and trade offs between different damage types and repairing methods will be discussed in the end.

II. Network Faults

A. Base model

This experiment is conducted on a fully connected two-layer model consists of an input layer, a hidden layer, and an output layer. To inject faults into the network, we selectively damage the hidden neurons after training. All models in this study are build similarly, with only differences in the amount of neurons in the hidden layer. We will discuss the results of them in the following sections.

B. Neuron damage

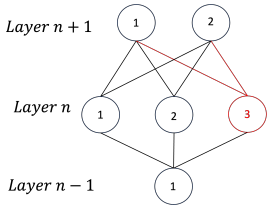


Fig. 1: Network with 33% neuron damage probability

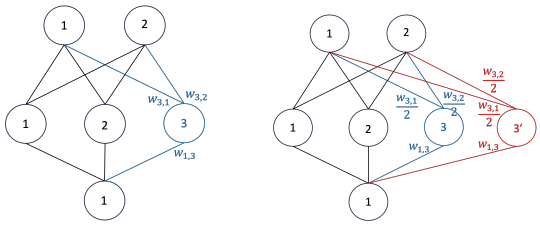
Neuron damage refers to the situation where the output of the faulty neuron is incorrect. Three different outcomes are considered in this experiment. The first and second are stuck-at faults where the neuron only outputs the minimum and maximum values respectively. In the third approach, damaged neurons output random noise subject to Eq. 1 below.

$$Noise = noise\_level \times \mathcal{N}(0,1) \tag{1}$$

Similar to [5], all damages on the neural network are evaluated by the damage probability in terms of accuracy. Fig. 1 illustrates an example of 33% neuron damage on the nth layer. The output of the neuron 3 in layer n is replaced with the noise in Eq. 1. Note that the damaged output acts as a faulty input to neuron 1 and neuron 2 in layer $n+1$. These two input values are the same as they were produced from the same faulty neuron in the previous layer.

III. Damage Repairing

A. Replicate neurons



(a) Original weights of the model

(b) Weights when neuron 3 is replicated once

Fig. 2: Example of duplicating a neuron once

In the first experiment, we exploit the fact that neuron outputs are the weighted sum of its inputs. Similar to [2],

augmentation is utilized to improve the fault tolerance of the model. After training, the network is explicitly changed to replicate hidden neurons while halving their weights accordingly. In this case, the information that each original neuron carries is distributed evenly between its replicated selves, thus allowing some information to be retained even if neuron damage occur. Fig. 2 demonstrates the weights before and after duplicating a specific neuron.

B. Modify the training process

For the second method, faults are purposefully injected into the training process in order to improve the fault tolerance of the model. The idea is similar to inserting dropout layers into neural networks, though instead of resetting the outputs to 0, we insert random noise subject to Eq. 1. The noise level is set to 1 in this experiment, thus making it equivalent to randomly injecting Gaussian noise during training. Note that when referring to injecting noise, the original output is replaced with noise instead of adding on top.

IV. Results

A. Neuron replication under stuck-at faults

In this section, we discuss the performance and observations when applying augmentation to our model after training.

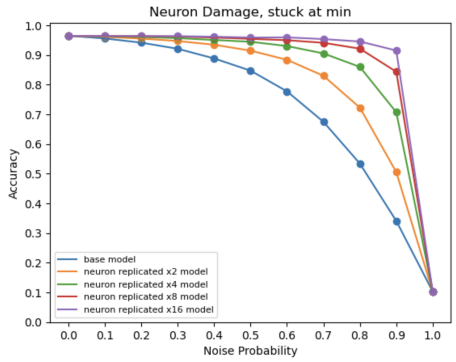


Fig. 3: Noise probability vs Accuracy with stuck-at minimum neuron damage

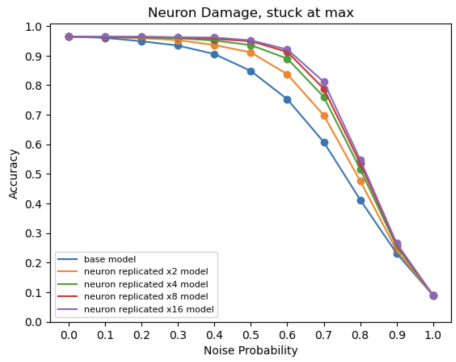


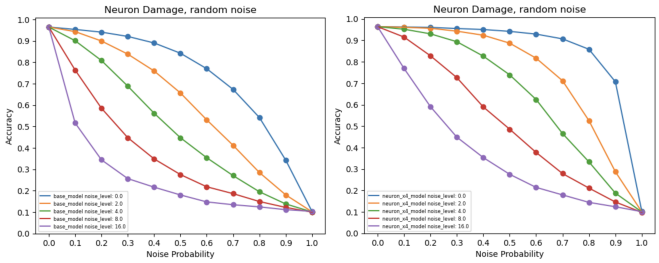
Fig. 4: Noise probability vs Accuracy with stuck-at maximum neuron damage

Fig. 3 and Fig. 4 represent the performance of the augmentation method under neuron damage during stuck-at faults. It can be observed that the accuracy drops at a slower rate when neurons are replicated compared to the base model, and the more replicants of each neuron, the better the model fault tolerance is. In cases such as when each neuron is duplicated 16 times with each replicant holding 1/16 of the original weight, the model can still reach over 90% accuracy when 90% of the neurons are damaged and stuck at minimum.

Note that augmentation works much better when neurons are stuck at minimum, with steady increases in fault tolerance as the replication counts rises. However, even though there are still subtle improvements, augmentation works much worse under stuck at maximum damages. This is possibly due to the fact that most neuron outputs before damage are closer to minimum, thus when stuck-at minimum faults occur, the relationships between the output of each neuron tend to be closer to the original when replicate counts are higher. On the other hand, stuck-at maximum faults will significantly alter the relationships of the output, as it may cause those originally insignificant neuron outputs, i.e. those that have values closer to minimum, to contribute more towards the input of the next layer.

B. Neuron replication under random noise damage

Under this section, the accuracy of the augmented model under random noise neuron damage is discussed. The experiment are conducted on different noise levels, while the random noises are created in accordance to Eq. 1.



(a) baseline model (b) replicated neuron model (x4)

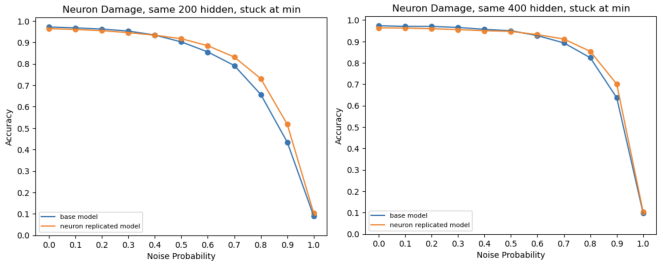
Fig. 5: Noise probability vs Accuracy with different noise levels of neuron damage

Fig. 5 depicts similar results to stuck-at faults. Regardless of the noise level, neuron replication provides a higher fault tolerance when compared to the original model. Replicating neurons provides a significant benefit to the model's fault tolerance under low noise levels, providing a performance boost of over 20% in some cases. Even though the accuracy decreases at a much faster rate with higher noise levels, augmentation can still provide subtle improvement towards the performance of the model.

C. Redundancy analysis of neuron replication

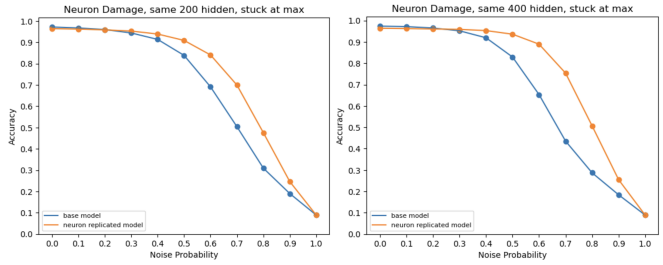
Augmentation and neuron replication trades off redundancy with fault tolerance. Interestingly, such redundancy would not

be honored during backpropagation, which is also observed in [2]. Accuracy increase under neuron damage is subtle even as hidden neurons are doubled. In this section, the experiment is conducted to showcase that under the same space constraints, augmentation provides a much higher fault tolerance improvement to the model compared to backpropagation.



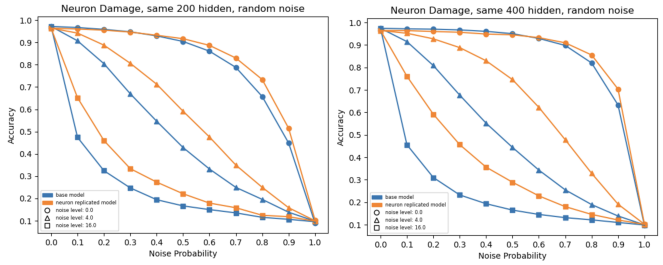
(a) 200 hidden neurons, stuck-at min

(b) 400 hidden neurons, stuck-at min



(c) 200 hidden neurons, stuck-at max

(d) 400 hidden neurons, stuck-at max



(e) 200 hidden neurons, random noise

(f) 400 hidden neurons, random noise

Fig. 6: Noise probability vs Accuracy under neuron damage with same number of hidden neurons

Fig. 6 showcases the final results. The experiment is conducted on all three neuron damage types. Note that both the base model and the replicated models consist of the same number of hidden neurons. For Fig. 6a, Fig. 6c, and Fig. 6e, the base model is trained with 200 hidden neurons, while the neuron replicated model applies augmentation on a model trained with 100 hidden neurons to duplicated each neuron once. Similar to the above, Fig. 6b, Fig. 6d, and Fig. 6f demonstrate the performance comparison of a base model trained with 400 hidden neurons and an augmented model that was trained with 100 hidden neurons and duplicated 4 times.

The above results indicate clearly that backpropagation takes little advantage of the redundant neurons. The accuracy improvement under various noise probabilities for the base model is minimal even if it is trained with more hidden neurons. The only obvious increase that can be observed is with stuck-at minimum damage. On the other hand, replicated neuron models show a much higher fault tolerance regardless of the damage type under the same space constraints. In addition, similar to conclusions raised from previous sections, augmentation can take advantage of the redundant neurons, providing a higher fault tolerance when more hidden neurons are allowed.

This phenomenon may be because the base model still converges to relying on a few critical neurons. When a damage occurs on one of these neurons, the model then tends to under perform. In contrast, neuron augmentation forces the model to distribute all neurons outputs more evenly, thus preventing the failure of a single or some critical points.

D. Modified training under stuck-at faults

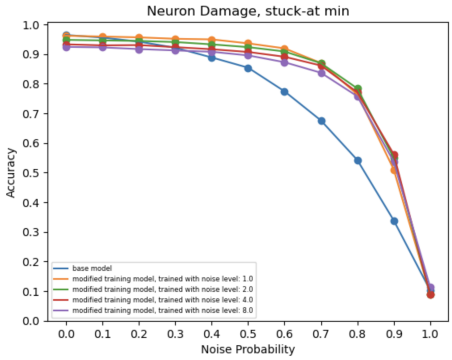


Fig. 7: Noise probability vs Accuracy with stuck-at minimum neuron damage with modified training model

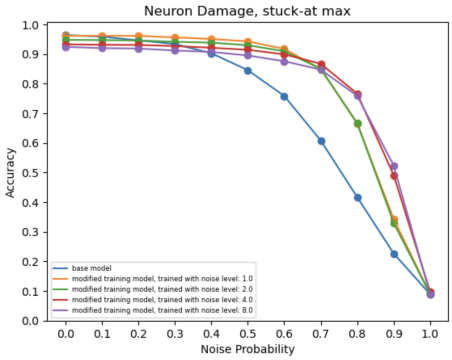


Fig. 8: Noise probability vs Accuracy with stuck-at maximum neuron damage with modified training model

One may ask if it would be possible to increase fault tolerance without trading off redundancy. In this section, instead of adding redundancy, we modify the training process to provide passive fault tolerance to the model. The experiment is conducted by injecting random noise following Eq. 1 while training the model. Different noise levels were tested and the

results are presented in Fig. 7 and Fig. 8. Unlike augmentation, the modified training models are not altered once the training process concludes.

Fig. 7 and Fig. 8 illustrate the results of modifying the training process with different noise levels under stuck-at faults. As shown in the figures, modifying the training process also improves the fault tolerance of the model significantly, especially when the model is heavily damaged. Interestingly, injecting noise with different noise levels during training do not impact its fault tolerance much. All modified training models seem to perform similarly regardless of the noise level they are trained with. Nevertheless, they all perform much better compared to the base model that is trained normally. Furthermore, the results also showcase that even when injecting random noise while training, the model can still prove to be fault tolerant towards stuck-at faults.

E. Modified training under random noise damage

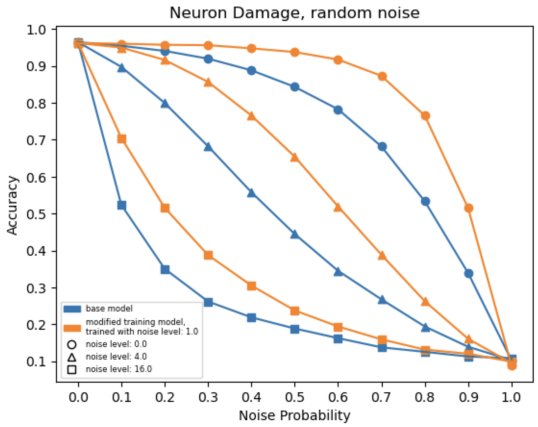


Fig. 9: Noise probability vs Accuracy with different noise levels of neuron damage with modified training model

Fig. 9 compares the base model with the modified training model that is trained with noise level 1.0. Similar to stuck-at faults, the modified training model provides a much higher fault tolerance regardless of the noise level damage. Interestingly, injecting noise with noise level 1.0 during training can still increase fault tolerance even when the model is damaged under random noise with a much higher noise level of 16.0.

V. Conclusion

In this study, we conducted experiments on inducing damage and enhancing fault tolerance of neural networks. The primary focus was on neuron damage categorized into three types: stuck-at minimum, stuck-at maximum, and random noise faults. Two methods were introduced to mitigate these damages and increase model accuracy. With neuron replication and augmentation, trade offs were made between redundancy and fault tolerance. Our results demonstrate that augmentation substantially enhances model performance across all damage types, and the improvement increases with more redundancy. Furthermore, a brief analysis showcased that under the same space constraints, while backpropagation fails to leverage the added redundancy, neuron augmentation effectively improves fault tolerance. We also explored an alternative method involving modifications to the training process, which does not require additional neurons. This technique also shows significant increase in accuracy under damage compared to the base model, allowing one to improve performance without additional space needed. These findings offer fresh insights into modifying the architecture or training process of the model and provide new perspectives when considering fault tolerance in neural networks.

References

[1] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[2] M. Emmerson and R. Damper, "Determining and improving the fault tolerance of multilayer perceptrons in a pattern-recognition application," *IEEE Transactions on Neural Networks*, vol. 4, no. 5, pp. 788–793, 1993.

[3] C. Sequin and R. Clay, "Fault tolerance in artificial neural networks," in *1990 IJCNN International Joint Conference on Neural Networks*, 1990, pp. 703–708 vol.1.

[4] C. Torres-Huitzil and B. Girau, "Fault and error tolerance in neural networks: A review," *IEEE Access*, vol. 5, pp. 17 322–17 341, 2017.

[5] S.-W. Chen, B. Duke, and P. Aarabi, "Faulty neural networks," in *2023 IEEE Conference on Artificial Intelligence (CAI)*, 2023, pp. 290–291.