Machine Learning and Intelligent Systems
J.-L. Kim (Ed.)

doi:10.3233/FAIA241216

Antimatter Networks for Combating the Dying ReLU Problem

Nick Sen CHUA [a] Tomas MAUL [b]
[a] *Mont' Kiara International School, Malaysia*
[b] *University of Nottingham Malaysia, Malaysia*

ORCiD ID: Nick Sen Chua https://orcid.org/0009-0004-1779-035X, Tomas Maul https://orcid.org/0000-0002-3563-1115

Abstract.

ReLU nodes are utilized commonly in neural networks as they look and act like linear functions while providing nonlinearity. In spite of addressing the vanishing gradient problem, they can lead to the dying ReLU problem which can be detrimental in terms of convergence and generalization performance. This paper proposes antimatter networks, a new and simple solution to the dying ReLU problem which involves combining ReLU nodes with their inverse, negative ReLU nodes, together with activation swapping mechanisms. We tested the solution on six separate dataset-architecture combinations with the MNIST, CIFAR-10, and Flowers-16 datasets for Convolutional Neural Networks (CNN) and Multi-Layer Perceptron networks (MLP) and found that antimatter networks lead to consistent convergence and generalization improvements compared to networks solely consisting of ReLU or NReLU nodes.

Keywords. Antimatter Network, Rectified Linear Unit (ReLU), Negative Rectified Linear Unit (NReLU), Multiple Activation Functions, Activation Swapping.

1. Introduction

A problem arises with utilizing ReLU activation functions ($f(x) = max(0,x)$), where if a neuron in the network has a weighted sum less than or equal to zero for all input instances, it will be perpetually inactive and thus cease to contribute to the learning process, since all gradients backpropagating through the node will be multiplied by zero. This phenomenon is commonly known as the dying ReLU problem. We propose a novel solution to this problem: combining ReLU nodes with their inverse, namely the negative rectified linear unit (NReLU), which is defined by $f(x) = min(0,x)$, together with different activation swapping mechanisms. Inspired by the complex relationship between matter and antimatter in the universe, we propose the coining of a new informal term for neural networks that primarily combine both ReLU and NReLU nodes, i.e.: antimatter networks. Although the exclusive combination of ReLU and NReLU activation functions has been explored before [1], to the best of our knowledge it has never been combined with swapping mechanisms [2] and a multi-branch architecture.

When one network uses both activation functions, the possibility for one function compensating for the other is created. In this paper, we focus on activation swapping in a two-branch architecture, where one branch consists of ReLU nodes, and the other consists of NReLU nodes. A two-branch architecture allows for a smaller number of weights (compared to an equivalent single-branch architecture with the same number of nodes) and also simpler and hence more efficient implementations of swapping mechanisms. It is important to note that these branches do not share weights as in a Siamese network, but rather can learn different representations. This paper proposes three different activation swapping variants as outlined in the methods section. The general hypothesis is that networks that combine ReLU and NReLU nodes, and implement an effective activation swapping mechanism, should demonstrate improved convergence and generalization properties (e.g. faster and more consistent convergence, and better and more consistent test accuracy), compared to baseline conditions. In this paper we equate faster convergence to lower mean training loss, and better generalization to higher mean test accuracy, at the end of training. Moreover, we use improved-convergence as a proxy for tackling the dying ReLU problem, since the number of dead ReLUs is generally positively correlated with the training loss (i.e. the more dead ReLU nodes a network exhibits, the harder it is for the network to converge).

2. Related Works

The key concept underlying the proposed solution pertains to the flexible adoption of multiple activation functions in the same network (or layer), rather than the exclusive adoption of a single activation function. This concept, although not mainstream, has a long history, and has appeared in the literature under different terms, e.g.: mixture of activation function networks [3], adaptive activation functions [4], neural diversity machines [5], activation ensembles [6], and others. As such, the proposed solution can be defined as a special case within this broader category, where we focus exclusively on ReLU and NReLU functions, and employ mechanisms that attempt to address limitations of ReLU-only networks (e.g. dead ReLU problem). It should be noted that many variations of the ReLU activation function (e.g. Leaky ReLU) are also useful for addressing the dead ReLU problem, however, these variations are not the focus of this paper.

The concept of combining both ReLU and NReLU nodes in the same network, although very rare, has some precedence in the literature. For example, the work in [1] as briefly reviewed in [7], reports on an activation function solution, named concatenated ReLU (CReLU), that doubles each node into two portions, corresponding to both positive and negative linear responses (i.e. $f(x) = (ReLU(x), ReLU(-x))$). The work in [1] demonstrated clear improvements in recognition performance for different convolutional neural network (CNN) architectures incorporating CReLU, on CIFAR-10, CIFAR-100 and ImageNet datasets. In spite of this similarity with our work, a crucial difference pertains to the absence of activation swapping mechanisms in [1], which we explore in some detail, and the fact that we incorporate both activation functions via a multi-branch architecture.

Although the notion of activation swapping or weight exchange can be found in distributed computing [8] and evolutionary contexts [9], to the best of our knowledge activation swapping has not been employed as an integral part of the learning process focusing on activation function concerns. The closest mechanism we have found to this

pertains to the swapping of node activations, which was explored in [2] under the name swap-node. The authors explored swap-node in CNNs, with a regularization motivation similar to dropout. However, our work distinguishes itself by the architecture and procedures within which we implement the concept (for the sake of clarity, we will refer to our procedure "activation swapping"), the set of conditions it uses to modulate its operation (e.g. based on inactivity levels), and its motivation (i.e. addressing the dead ReLU problem).

3. Methodology

3.1. Activation Swapping Mechanisms

Three variants of activation swapping, applied at each epoch, were experimented with: (1) swap the most inactive ReLU node and the most inactive NReLU node's values, (2) swap all ReLU and NReLU nodes' values that are inactive for more than 70% of data instances, and (3) swap a random pair of ReLU-NReLU nodes' values probabilistically in a manner that is directly proportional to their inactivity levels. We define the "most inactive node" as the node that is inactive for the largest number of data instances.

3.2. Datasets and Performance Metrics

For evaluation purposes three image-based classification datasets were used, namely MNIST, CIFAR-10, and Flowers-16. To measure generalization performance the means and standard deviations of test accuracies were computed for different experimental conditions. To measure convergence performance, the mean and standard deviation of the final training cross-entropy loss were computed for the same conditions.

3.3. Architectures and Training

The reported experiments utilized two control architectures per dataset: a standard multilayer perceptron (MLP) and a standard convolutional neural network (CNN) with single activation functions. Both backbones adopted a simple two-branch design for baseline conditions and were modified to incorporate the antimatter concept for the experimental conditions.

For the MLP control network with the MNIST dataset, we used two branches, both of which took input images through the first layer (28x28) and then processed the images into 128 nodes for the first hidden layer, to which the activation function was then applied (ReLU (C1), NReLU (C2), or both (C3-C6; with ReLU in one branch, and NReLU in the other). Then, the first hidden layer processes those 128 nodes into 32 nodes, applies the same activation function again, which leads to a shared dense block whose input layer takes in 64 nodes (32 from the first branch and 32 from the second branch), whose output consists of the final ten nodes, to which is applied a log softmax function to determine the final output.

For the CNN network, first a convolutional layer transforms the input to 32 channels, to which the activation function of choice is then applied, and another convolutional layer increases the channel to 64 channels. An activation function follows, followed by a max

pooling layer that reduces the spatial dimensions to 14x14, and another convolutional layer increases the channels to 128. Another activation function is applied; a max pooling layer is used to reduce the spatial dimensions to 7x7, and another convolutional layer is used to increase the channels to 256 with an activation function following. A final max pooling layer reduces the spatial dimensions to 3x3; the layer gets flattened (256 x 3 x 3) and is inputted into a fully connected layer to reduce the number of nodes to 512. Finally, an activation function is applied to those 512 nodes, which gets inputted into another layer, which reduces the number of nodes further to 256, and an activation function is applied. The 256 nodes lead to a shared block that processes the input of 512 (256 nodes from the first branch and 256 nodes from the second branch) to 10 nodes (for determining the output).

As for the training of our MLP and CNN networks, our MLP network was trained over the course of 15 epochs, using a negative log-likelihood (NLL) loss function and stochastic gradient descent (SDG) with a learning rate of 3×10^{-3} and momentum of 0.9. On the other hand, our CNN network was trained across 7 epochs, using a cross-entropy loss function and Adam optimizer. Finally, no instances of dropout or other regularization techniques were used in training our MLP or CNN networks as the primary goal was to evaluate the core architectures and learning mechanisms without the influence of additional constraints.

3.4. Experimental Design

The experimental design aims to answer the research question mentioned in the introduction: can one or more variants of antimatter networks be shown to perform better than baseline models, where performance is measured in terms of convergence and generalization properties? In other words, are any antimatter network variants superior to common single-activation function networks in one or more of the adopted performance metrics?

We developed several variations of antimatter networks and control groups to address the question. All conditions use a two-branch architecture. Both control conditions adopt a single activation function, C1 with ReLU, and C2 with NReLU. Condition C3 combines both ReLU and NReLU (one in each branch) and does not use any swapping mechanism, as a control antimatter network. Each branch implements non-linear transformations according to its adopted activation function (ReLU or NReLU), and these non-linear mappings are then combined by the final linear layer (followed by a softmax function). Conditions C4-C6 employ both ReLU and NReLU activation functions (like C3), but employ different activation swapping mechanisms. C4 swapping is done by identifying the single most inactive ReLU node, and the single most inactive NReLU node, and swapping their activations. C5 swapping is done by identifying which ReLU and NRelU nodes are inactive for more than 70% of instances, and then swapping pairs of corresponding nodes. C6 swapping is done by selecting a ReLU-NReLU pair of nodes probabilistically, in a way that is directly proportional to their level of inactivity (i.e. the more inactive a node is, the larger the probability of it being selected), and then swapping activations. For each specific condition and combination, 10 experimental trials were repeated to make sure the results were consistent across different trials and provided a robust basis for concluding the effectiveness of our antimatter network configurations.

4. Results

Table 1.Mean test accuracy across all conditions with the standard deviation in brackets. C1: ReLU. C2: NReLU. C3: ReLU-NReLU no swapping. C4: ReLU-NReLU swap most inactive. C5: ReLU-NReLU swap $>$ 70% inactive. C6: ReLU-NReLU swap probabilistically.

	Dataset - Architecture Combination					
Condition	**MNIST MLP**	**MNIST CNN**	**CIFAR-10 MLP**	**CIFAR-10 CNN**	**Flowers-16 MLP**	**Flowers-16 CNN**
C1	97.21% (0.18)	99.05% (0.11)	50.58% (0.38)	76.53% (0.96)	44.75% (1.52)	58.91% (3.76)
C2	97.06% (0.16)	98.77% (0.33)	50.62% (0.28)	69.84% (0.79)	44.67% (1.67)	61.02% (2.39)
C3	**97.24% (0.13)**	**99.23% (0.10)**	49.06% (0.26)	77.23% (0.38)	44.94% (1.38)	**61.33% (2.03)**
C4	97.23% (0.15)	98.96% (0.13)	**50.85% (0.22)**	77.34% (0.27)	44.89% (0.91)	60.43% (4.49)
C5	97.17% (0.17)	99.0% (0.20)	50.79% (0.23)	**78.11% (0.67)**	**45.15% (1.67)**	58.66% (2.79)
C6	97.13% (0.14)	99.09% (0.16)	50.72% (0.40)	75.63% (0.65)	45.09% (1.60)	59.79% (2.84)

The first section of this experiment compares the performance of antimatter networks in general (C3-C6) against baseline conditions (C1 and C2), in terms of test accuracy. The results, as depicted in Table 1 provide strong evidence in support of the favorable generalization properties of antimatter networks. In all six architecture-dataset combinations the best performer was always an antimatter condition (i.e. C3, C4 or C5).

Next, regarding convergence, as shown in Table 2, antimatter networks (i.e. C3-C6) have consistently converged better than the non-antimatter control cases with ReLU and NReLU alone (i.e. C1 and C2). In spite of convergence performance indicating a positive result for the antimatter concept, suggesting a partial addressing of the dying ReLU issue, the results were not as consistent compared to the generalization results, indicating that more work is required to refine the techniques.

Table 2.Mean and standard deviation of the final training losses across various architectures and datasets. C1: ReLU. C2 NReLU. C3: ReLU-NReLU no swapping. C4: ReLU-NReLU swap most inactive. C5: ReLU-NReLU swap $>$ 70% inactive. C6: ReLU-NReLU swap probabilistically.

	Dataset - Architecture Combination					
Condition	**MNIST MLP**	**MNIST CNN**	**CIFAR-10 MLP**	**CIFAR-10 CNN**	**Flowers-16 MLP**	**Flowers-16 CNN**
C1	0.602 (0.0021)	0.0211 (0.0012)	1.431 (0.0028)	0.3721 (0.0278)	1.7404 (0.0157)	**1.2767 (0.0708)**
C2	**0.06 (0.0015)**	0.0254 (0.002)	1.4293 (0.0028)	0.5922 (0.039)	1.7412 (0.0271)	1.4285 (0.0763)
C3	0.0606 (0.0017)	**0.0206 (0.0027)**	**1.427 (0.0026)**	0.0807 (0.1002)	1.7328 (0.0262)	1.3183 (0.0454)
C4	0.0613 (0.0015)	0.0232 (0.0019)	1.4285 (0.0061)	0.0805 (0.0023)	1.7357 (0.0053)	1.3435 (0.0056)
C5	0.0602 (0.0016)	0.022 (0.0018)	1.4276 (0.0064)	**0.0784 (0.0021)**	1.7590 (0.0057)	1.3597 (0.0062)
C6	0.0611 (0.0015)	0.0236 (0.0022)	1.4298 (0.0063)	0.0817 (0.0022)	**1.7217 (0.0056)**	1.3584 (0.0061)

The second section of this experiment aims to discover whether antimatter activation swapping variants (i.e. C4, C5 and C6) generally outperform non-antimatter networks (C1 and C2), or default antimatter networks with no activation swapping (i.e. C3). According to Table 1, it is clear that there were slight improvements for some dataset-architecture variants on certain conditions. However, although swapping mechanisms showed their promise, these effects were relatively variable and data-architecture specific, suggesting that new activation and/or weight swapping mechanisms should be explored in the future.

Finally, as shown in Table 2, our activation swapping variants only converged the best in two of the six different dataset-architecture combinations. Overall, this indicates that antimatter activation swapping conditions require further work to improve convergence properties.

5. Discussion

The results demonstrate that antimatter networks produce consistent performance improvements compared to networks with single activation functions. Whether through activation swapping (i.e. C4-C6) or the variant without activation swapping (i.e. C3), there is a general increase in mean test accuracy and decrease in the corresponding standard deviation compared to single activation function conditions (i.e. C1 and C2), which means generalization is better and more reliable in antimatter networks. Furthermore, our C3 condition, in most cases, performs better in terms of convergence, as measured by the final training loss, in comparison to C1 and C2, proving that antimatter networks can generally lead to convergence improvements, however this result is less consistent in the case of activation swapping conditions, which implies that future work should continue to explore how different mechanisms can be used to improve the synergy between ReLU and NReLU nodes.

If we take a higher-level perspective and aggregate the conditions into two categories, i.e. non-antimatter conditions (i.e. C1 and C2) and antimatter conditions (i.e. C3-C6), and compare these two categories in terms of the proportion of architecture-dataset combinations obtaining the best mean performance, we find a strong advantage for the antimatter category. For convergence properties, as measured by the mean final training loss (Table 2), the antimatter category exhibited best performance in 66.6% of cases, whereas for generalization properties (Table 1), it exhibited best performance in 100% of cases. This broader perspective clearly demonstrates that the combination of ReLU and NReLU nodes in the same architecture can have a reliable positive effect on both convergence and generalization, across different architectures and datasets. The positive convergence effect is suggestive that the antimatter architecture has a positive effect on the dying ReLU problem, however additional experiments are required to measure the effect more directly.

Future work should investigate a broader range of architectures through a more flexible architectural design space with extensive hyperparameter tuning, and a more extensive array of swapping mechanisms. This would allow one to draw stronger conclusions regarding the robustness and general applicability of the approach. Moreover, a broader spectrum of performance metrics, including those involving continual learning and multi-

task learning, should provide additional insights into the strengths and limitations of the antimatter concept.

References

[1] Wenling Shang et al. "Understanding and improving convolutional neural networks via concatenated rectified linear units". In: *International conference on machine learning*. PMLR. 2016, pp. 2217–2225.

[2] Takayoshi Yamashita et al. "SWAP-NODE: A regularization approach for deep convolutional neural networks". In: *2015 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2015, pp. 2475–2479.

[3] Alexander Hagg, Maximilian Mensing, and Alexander Asteroth. "Evolving parsimonious networks by mixing activation functions". In: *Proceedings of the genetic and evolutionary computation conference*. 2017, pp. 425–432.

[4] Sheng Qian et al. "Adaptive activation functions in convolutional neural networks". In: *Neurocomputing* 272 (2018), pp. 204–212.

[5] Tomas Maul. "Early experiments with neural diversity machines". In: *Neurocomputing* 113 (2013), pp. 36–48.

[6] Diego Klabjan and Mark Harmon. "Activation ensembles for deep neural networks". In: *2019 IEEE International Conference on Big Data (Big Data)*. IEEE. 2019, pp. 206–214.

[7] Vladimír Kunc and Jiří Kléma. "Three Decades of Activations: A Comprehensive Survey of 400 Activation Functions for Neural Networks". In: *arXiv preprint arXiv:2402.09092* (2024).

[8] Julian Dorner, Samuel Favrichon, and Arif Selcuk Ogrenci. "Weight exchange in distributed learning". In: *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2016, pp. 3081–3084.

[9] Dario Floreano, Peter Dürr, and Claudio Mattiussi. "Neuroevolution: from architectures to learning". In: *Evolutionary intelligence* 1 (2008), pp. 47–62.