Accurate Neuron Resilience Prediction for a Flexible Reliability Management in Neural Network Accelerators

Christoph Schorn*†, Andre Guntoro†, Gerd Ascheid*
*Institute for Communication Technologies and Embedded Systems, RWTH Aachen University, Germany
†Robert Bosch GmbH, Corporate Research, Renningen, Germany
Email: christoph.schorn@de.bosch.com, andre.guntoro@de.bosch.com, gerd.ascheid@ice.rwth-aachen.de

***Abstract*—Deep neural networks have become a ubiquitous tool for mastering complex classification tasks. Current research focuses on the development of power-efficient and fast neural network hardware accelerators for mobile and embedded devices. However, when used in safety-critical applications, for example autonomously operating vehicles, the reliability of such accelerators becomes a further optimization criterion which can stand in contrast to power-efficiency and latency. Furthermore, ensuring hardware reliability becomes increasingly challenging for shrinking structure widths and rising power densities in the nanometer semiconductor technology era. One solution to this challenge is the exploitation of fault tolerant parts in deep neural networks. In this paper we propose a new method for predicting the error resilience of neurons in deep neural networks and show that this method significantly improves upon existing methods in terms of accuracy as well as interpretability. We evaluate prediction accuracy by simulating hardware faults in networks trained on the CIFAR-10 and ILSVRC image classification benchmarks and protecting neurons according to the resilience estimations. In addition, we demonstrate how our resilience prediction can be used for a flexible trade-off between reliability and efficiency in neural network hardware accelerators.**

I. Introduction

Deep neural networks (DNNs) have triggered a revolution in computer vision and effectively become the predominant approach for most visual recognition and detection tasks [1]. With DNN-based methods surpassing human-level performance in some very complex decision problems (e.g. [2]–[4]), they are developing into a key component for the environmental perception in open context systems, for example autonomously driving vehicles. However, the application of neural networks in cars poses some challenges on the design of future automotive hardware platforms. On the one hand, efficient and fast accelerators are needed, since state-of-the-art DNNs for image recognition exhibit massive computational requirements [5]. On the other hand, functional safety regulations (e. g. ISO 26262 [6]) demand redundant designs to ensure fault tolerance of critical hardware components, which will be even more important in the case of self-driving cars. At the same time, ensuring reliability becomes increasingly challenging in modern integrated circuit technology [7]. The reason for this is the ongoing trend towards shrinking structure widths, which raises the circuit susceptibility, for example to electrical noise, aging, process and temperature variations as well as radiation-induced soft errors [8], [9]. In addition, emerging low-power techniques, such as near threshold voltage computing and approximate computing, further complicate reliability management [10].

To overcome these challenges, one option is to exploit fault tolerance on the algorithm level and allow for a certain degree of inaccuracy on the hardware level. Several studies have analyzed the fault tolerance of neural networks and shown that they can be designed and trained to be highly robust against computation errors [11], [12]. Since individual network parts differ in their error resilience, an accurate and fine-grained resilience characterization method is needed for deciding which computations have to be safeguarded against errors and which can be executed on potentially unreliable or approximate hardware elements. We focus on analytical resilience estimation methods, as they are interpretable and faster than simulation-based approaches for modern large-scale DNNs. Nonetheless, fault simulations are employed to test the analytical methods.

We contribute to research in three ways. Firstly, we suggest a new approach for an accurate resilience prediction on the individual neuron level, which is founded on the recently introduced deep Taylor decomposition of neural networks [13]. Secondly, we propose a framework for a resilience-based flexible reliability management in reconfigurable neural network accelerators. And thirdly, we compare our method to state-of-the-art resilience prediction techniques by performing fault injection simulations with two DNN image classifiers and two different fault models. We extend this comparison by a detailed difference analysis of the methods.

II. Related Work

The understanding of how DNNs make their classification decisions has been the focus of various recent publications [13]–[17]. This stands in close relation to our research, as it is a necessary prerequisite for assigning accurate resilience values to individual neurons of the network. Our method builds on top of the recently proposed deep Taylor decomposition by [13]. The authors of this study derived a set of rules for determining the pixelwise contribution of the input image to the classification decision. This is done by propagating the relevance for a given target layerwise through the network

978-3-9819263-0-9/DATE18/©2018 EDAA

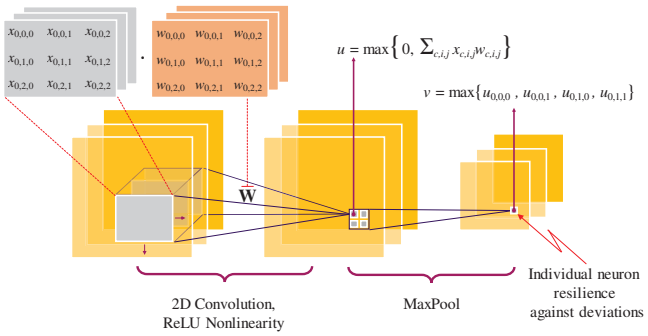


Fig. 1. Schematic depiction of the individual steps taking place in a convolutional layer function block.

back to the input. Our research differs from [13] as we employ the relevance propagation with a different intention. Instead of in a saliency map in the input space, we are interested in the intermediate relevance scores that are assigned to the neurons of each layer. Furthermore, we take the mean of these scores over a training dataset, since we are not focusing on individual classification decisions, but the global classifier behavior.

Resilience prediction on an individual neuron level has recently been proposed by two other groups of authors [18], [19]. Though the scope of their studies is slightly different, as they focus on approximating parts of the DNN, they pursue a very similar goal as we do, namely the optimal split up of a neural network's neurons to hardware units with different levels of exactness. Both use the same underlying technique, which is based on the backpropagation of error gradients, similar to neural network training. We will compare our own method with their gradient-based approach and highlight differences later in this paper.

III. Methodology

A. Preliminaries

A feed-forward DNN can be recursively defined as the concatenation of N layers, where the i^{th} layer computes its output based on the output of the preceding layer

$$\mathbf{y}_i = \mathbf{f}_i(\mathbf{y}_{i-1}). \tag{1}$$

The first layer input $\mathbf{y}_0$ represents the input data of the network, for example an image[1]. The last layer output $\mathbf{y}_N$ assigns a probability to each of the possible classes. In our definition each function $\mathbf{f}_i$ of a layer includes multiple operations. For the first layers of a convolutional neural network (CNN), it typically consists of a two-dimensional convolution of the input feature maps with a number of different weight kernels, followed by rectified linear units (ReLUs) [20] as nonlinearities. In some of the convolutional layers a pooling operation, for example maximum-pooling, is appended in order to reduce the spatial dimension. The individual operations of such a layer block are visualized in Fig. 1. The resulting output consists of multiple feature maps, which are the input to the next following layer. We call each pixel in these output feature maps a *neuron* and want to determine its corresponding error resilience.

The last few layers of the network before the final output are often realized as fully connected layers, which means that each neuron output is a weighted sum of all the previous layer outputs with subsequent nonlinearity. Typically, ReLU nonlinearities are applied here as well, except for the last layer, where the softmax function is used in order to output class probabilities.

B. Neuron Resilience Prediction

We characterize the error resilience of a network's individual neurons based on their average contribution to the target output of the DNN. The underlying hypothesis is that neurons which strongly contribute to the output of a neural network have a high impact on the classification accuracy degradation in case their output is disturbed. Consequently, a high average contribution means low error resilience and vice versa. As shown in [13], the contribution of each neuron to the output function value of a DNN can be determined by utilizing a Taylor decomposition and layerwise relevance propagation (LRP). The LRP algorithm iteratively propagates the contribution $R_{i+1,k}$ of each neuron k in the layer $i+1$ backwards to the neurons j of the previous layer. In the case of ReLU neurons the propagation rule is given by

$$R_{i,j} = \sum_k \frac{y_{i,j} \max(0, w_{j,k})}{\sum_{j'} y_{i,j'} \max(0, w_{j',k})} R_{i+1,k}, \tag{2}$$

where $y_{i,j}$ is the output value of the j^{th} neuron in the i^{th} layer for the given network input and $w_{j,k}$ is the weight which connects neuron j with neuron k. The initial propagation value $R_{N,k}$ of the last layer neurons is given by the one-hot encoded class label vector $\mathbf{t}$ belonging to the input image $\mathbf{y}_0$.

The function $R_{i,j}(\mathbf{y}_0, \mathbf{t})$ assigns a target output contribution score between 0 and 1 to neuron j. Since we are interested in the average contribution of each neuron, we take the mean over a set of M training images $\{\mathbf{y}_{0,0}, \mathbf{y}_{0,1}, \ldots, \mathbf{y}_{0,M-1}\}$ with associated labels $\{\mathbf{t}_0, \mathbf{t}_1, \ldots, \mathbf{t}_{M-1}\}$. We define the estimated resilience of each neuron $y_{i,j}$ based on this training set as

$$r_{i,j} = \frac{M}{\sum_{m=0}^{M-1} R_{i,j}(\mathbf{y}_{0,m}, \mathbf{t}_m)}. \tag{3}$$

According to (3), the resilience of neurons without any output contribution over the whole training set becomes infinite[2], which means that the output of these neurons can be set to zero without any effect on the classification accuracy, at least for the training set.

A remarkable property of our resilience definition is that the mean resilience over all J neurons of a given layer always equals the number of neurons in that layer

$$\frac{1}{J} \sum_{j=0}^{J-1} r_{i,j} = J. \tag{4}$$

[1] In the following, we focus on image classification networks, though our method is not limited to this domain.

[2] In practice, we set their resilience values to a sufficiently large number.

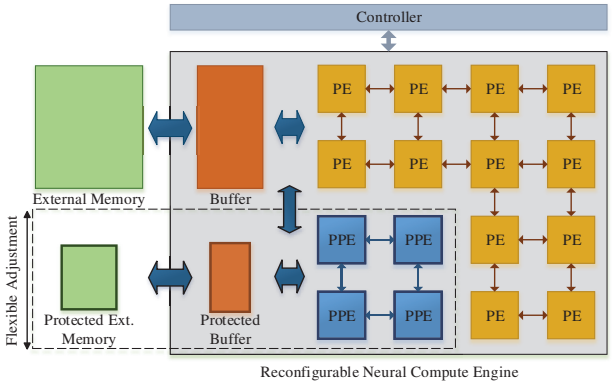


Fig. 2. Architecture of a Neural Compute Engine with flexible reliability management based on a neuron resilience ranking.

This property stems from the fact that LRP conserves the total relevance during the propagation from one layer to the next and that the initial relevance, given by the target vector $\mathbf{t}$, is one. It means that the mean layer resilience is proportional to the number of neurons in that layer, which is plausible, since in a layer with a larger number of neurons, each individual neuron carries on average a smaller fraction of the total relevant information.

C. Flexible Reliability Management

Based on the estimated neuron resilience values from (3), we compile a sorted list of all neurons of the network. This list represents a priority ranking, meaning that neurons with the lowest estimated resilience have the highest priority to be calculated on safeguarded hardware elements and vice versa. A high-level conceptual view of our proposed Neural Compute Engine is depicted in Fig. 2. The controller maps the fraction of neurons with the lowest resilience to protected processing elements (PPEs) and their intermediate and final computation results to protected local memory buffers or external memory. The protected hardware elements can be realized by adding spatial or temporal redundancy and by using error correction mechanisms. All other neuron computations and corresponding results are mapped to regular processing elements (PEs), which are potentially unreliable or less accurate on the one hand, but consume less energy and silicon area on the other hand. To minimize the risk of an undesired fault propagation, the protected region should be isolated as best as possible. Only on the data bus level between buffers and processing elements, a transition between the protected and the unreliable domain has to be allowed, because some PPEs require previous layer outputs from unreliable PEs as inputs and vice versa.

The proportion of protected and unreliable elements can be reconfigured either statically, at design time, or dynamically, at run-time, which allows for a flexible trade-off between efficiency and reliability. The target proportion can be determined for example based on a dynamic output quality or power objective, or based on some fault rate tracking within the unprotected hardware elements.

TABLE I
NETWORK ARCHITECTURES AND DATASETS

Architecture	Dataset	Layers	Neurons
All-CNN [21]	CIFAR-10 [22]	9 convolutional	357 002
VGG-16 [23]	ILSVRC-2012 classification [24]	13 convolutional, 3 fully connected	13 556 712

IV. EXPERIMENTS

A. Experimental Setup

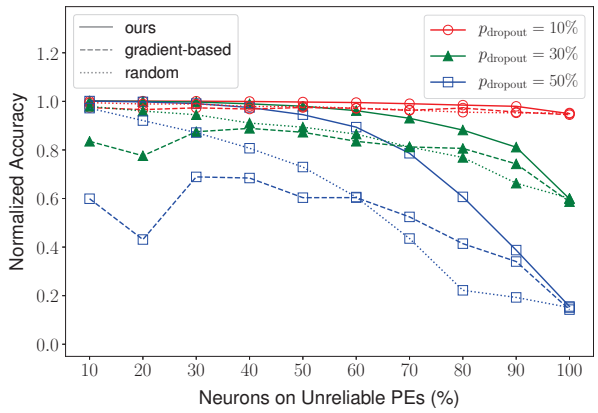
In order to evaluate our method, we estimate the neuron resilience values of two contemporary DNN image classifiers on two different benchmark datasets. The first network we consider is an All-CNN, as defined in [21], trained on CIFAR-10 [22], a dataset consisting of 32×32 pixel RGB images divided into ten different classes. Our implementation of the network classifies 88.07% of the test images correctly. The second network is the much larger VGG-16 architecture from [23]. It was trained on the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) classification dataset [24], which consists of 224×224 pixel RGB images from 1000 different classes. The classification accuracy of the network on the validation dataset is 68.55%. The properties of the two classifiers are summarized in Table I.

We use our own resilience prediction method and two other methods, which are described in the next section, for comparison. For the resilience computations, we use the complete CIFAR-10 training set of 50 000 images for the All-CNN, and a subset of 10 000 images from the ILSVRC training set for the VGG-16 network to speed up computation. The resulting resilience estimates of each method are used to create neuron resilience rankings. Then, different splits between PE and PPE computations are tested, while each time the lowest ranked neurons are mapped to PPEs. To evaluate the accuracy of the individual rankings, we simulate the effect of faults in PEs, considering two different fault models. The resilience prediction methods and fault injection simulations are implemented in our own extension of the Keras [25] deep learning framework, using the Theano [26] backend and an NVIDIA Tesla K80 GPU.

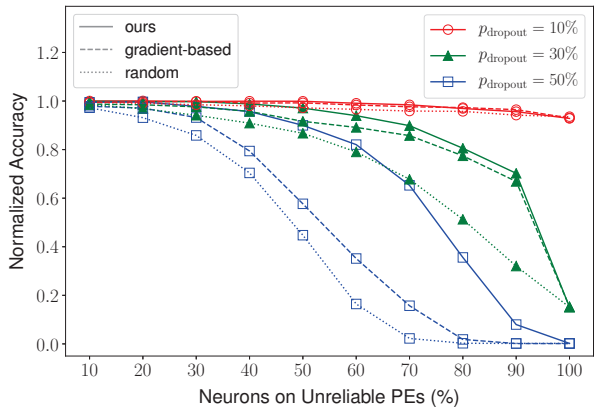
B. Other Resilience Prediction Methods

We compare our method to the gradient-based resilience prediction approach in [18] and [19]. To assign resilience values to the individual neurons, both references propose to use a modification of the error backpropagation algorithm, which is commonly used during neural network training. In our implementation of this method, we use the same training data for resilience prediction as for our own method.

For a second comparison we also take a random neuron resilience assignment into account. This method draws for each neuron a sample from the unit normal distribution and assigns the absolute value of it as resilience value. The resulting random neuron priority ranking serves as a baseline for the resilience prediction accuracy.

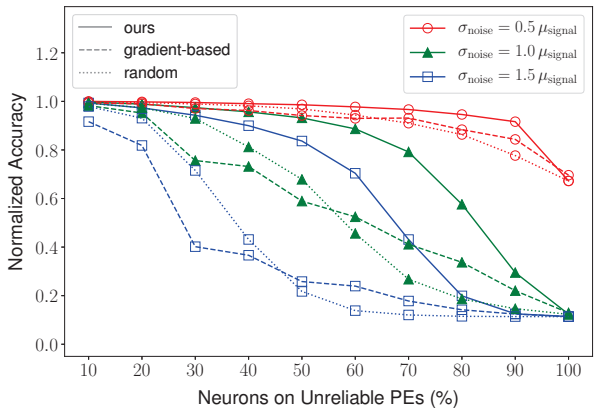


(a) All-CNN (CIFAR-10)

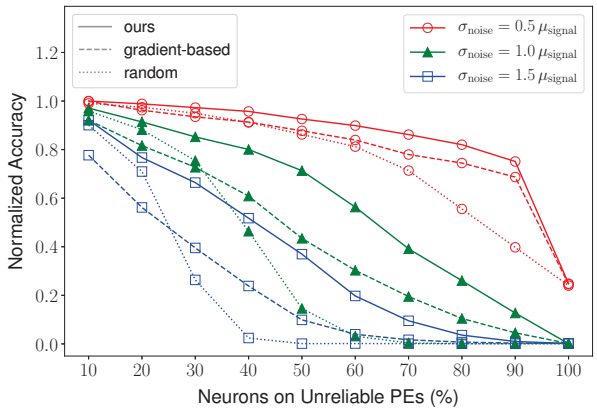


(b) VGG-16 (ILSVRC)

Fig. 3. Results of the fault injection simulations with the dropout fault model.



(a) All-CNN (CIFAR-10)



(b) VGG-16 (ILSVRC)

Fig. 4. Results of the fault injection simulations with the additive Gaussian noise model.

C. Dropout Fault Model

The first model we analyze is the dropout model in which a certain fraction of neuron outputs is set to zero. The assumption in this model is that each computation of a neuron on an unreliable PE independently fails with the probability p_{dropout} and in the failure case the PE outputs zero. There are various underlying causes which can lead to this error characteristic. It could for example be caused by the permanent or intermittent breakdown of some PEs, a fault in the interconnection between PEs and memory or in the memory controller. There could also be some fault detection logic in place, where the PEs enter an error condition with zero output as soon as a fault is detected. The model assumes that dropout only occurs in unreliable PEs, while PPEs and their associated memories are safeguarded, for example through the use of redundant computations and error correction techniques.

Fig. 3 shows the results of our simulations with the dropout model for the CIFAR-10 and the ILSVRC network and different dropout probabilities. For measuring the resulting accuracy of each network after fault injection, we let it classify a set of 10 000 test images, which is different from the training set that was used for the resilience prediction. To account for the stochasticity of the fault injection positions, we report the mean results over 10 iterations of the simulation. As can be seen, our method outperforms the two other methods in all cases (except for the trivial case of 100% unreliable PEs, where no resilience ranking is required). We can put for example up to 40% of the neuron computations on unprotected PEs with a large dropout probability of up to 50% without losing more than 5% accuracy. Interestingly, the gradient-based method exhibits a much weaker performance, with accuracy drops of more than 50% in the worst-case, and it is even partly outperformed by the random method for the All-CNN. Furthermore, it can be seen that accuracy degradation is not monotonous for the gradient-based approach, which indicates a wrong ordering in the predicted resilience ranking.

D. Additive Gaussian Noise Model

While the dropout fault model describes PEs which output zero, the additive Gaussian noise (AGN) model assumes that on each output of the unreliable PEs a random noise value is added, which is independently drawn from a Gaussian distribution for each neuron. Since each neuron output is composed of typically many multiply and accumulate operations, which are assumed to be independently affected by faults, the resulting

noise added to the neuron output can be approximated by a zero-mean Gaussian distribution according to the central limit theorem. This model is applicable to describe the effect of faults which happen on the level of the individual arithmetic operations within a PE as well as errors in the input values that are loaded from memory. This includes for example the effect of quantization errors and soft errors. To account for the different magnitudes of the neuron output values in different layers, we scale the σ_{noise} parameter of the Gaussian distribution in each layer with the corresponding layerwise mean neuron output value μ_{signal}.

The results of our simulations with the AGN model for the CIFAR-10 and the ILSVRC network are presented in Fig. 4. Again, we use a separate test set of 10 000 images and report the mean accuracy over 10 iterations of the simulation. Similarly to the dropout model, our LRP-based method also outperforms the other methods by significant margins in the AGN model and the gradient-based method is sometimes even outperformed by the random resilience assignment.

V. Discussion

A. Resilience Assignment Within a Layer

The results of the fault injection simulations suggest that our resilience prediction method generates a more accurate neuron resilience ranking than the gradient-based method. On that account, we will conduct a more detailed analysis of the differences between the two methods in the following. Fig. 5 gives an example of the resilience assignment to the first layer neurons of the VGG-16 network. Fig. 5b and Fig. 5c show the resilience values averaged over all output feature maps predicted by the LRP-based and the gradient-based method for a single sample image with the label "bee-eater" (Fig. 5a). Low resilience values (i. e. high relevancies) are displayed in dark color. It is apparent that the LRP method delivers a much more accurate indication of the features which are relevant for the correct classification of the image. This is consistent with the findings presented in [13]. By contrast, the gradient-based approach assigns a significant amount of relevance to neurons which are unrelated to the object to be classified. The overall effect of this on the resilience prediction can be observed in Fig. 5d, which shows the difference of the feature map mean resilience predicted by the gradient-based method and our LRP-based method, taking the whole training set of images into account. Since the gradient-based method spreads the relevance far more over the whole feature map, it *underestimates* the resilience of neurons at the margin. Similar results are obtained for the other convolutional layers of the VGG-16 network as well as for the All-CNN. We believe that the inaccurate spatial resilience assignment to the feature map neurons is one reason for the weak performance of the gradient-based method that we observed during our fault injection simulations.

B. Resilience Assignment Across Layers

Having discussed the differences of the methods within a single layer, we will now move on to analyze the inter-layer resilience assignment. Therefore we consider the mean resilience over all neurons per layer. As discussed earlier, our hypothesis is that the layerwise mean resilience should be proportional to the number of neurons in that layer, which is guaranteed by (4) for our method. In Fig. 6 we compare the mean layer resilience estimations of the three methods normalized by the number of neurons per layer. It is apparent that for our method the mean resilience equals the number of neurons in a layer, while the other two methods lack this proportionality. For the random resilience assignment this can be related to the fact that a random neuron resilience ranking leads to a constant mean resilience across layers and consequently the normalized mean resilience is inversely proportional to the number of neurons.

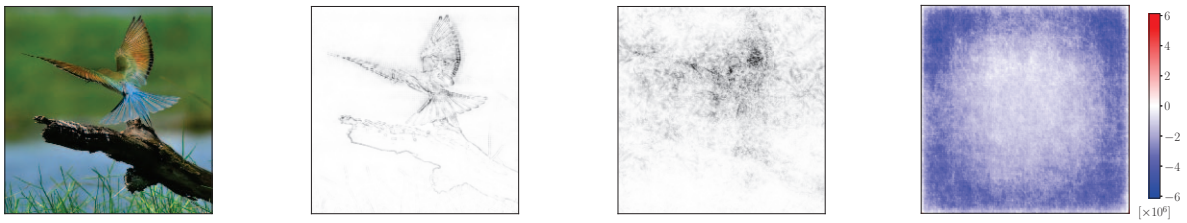
By contrast, the strong deviation between the gradient-based method and our method may partly be explained by the fact that the gradient-based method lacks a normalization of the error contributions which are assigned to the individual neurons through backpropagation. In fact, these attributions strongly depend on the training state of the network. A hypothetical perfect classifier would have a loss of zero over the training set and as a result the gradient-based method would assign an infinite resilience to all neurons of the network. However, it is obvious that not all neurons of the network can be set to zero without degradation of the output quality in that case. We believe that our method, which assigns a normalized resilience budget to the neurons of the network, therefore creates a more accurate resilience ranking. Moreover, we want to emphasize that this makes the individual neuron resilience assignments of our method comparable across different layers and even different network architectures, while this property is not given for the gradient-based method.

VI. Conclusion

We have proposed a method for predicting neural network error resilience on an individual neuron level. Our empirical findings with two current DNN image classifiers demonstrate the superiority of our method to previously published approaches. A detailed analysis revealed that our method results in a more accurate resilience assignment compared to the other methods. Consequently, we are able to shift parts of the neuron computations to unreliable hardware elements, while keeping the accuracy degradation of the network comparatively small. The accurate neuron resilience ranking which is generated by our method allows for trading off redundancy and efficiency in an optimal way. Thus, it adds a degree of flexibility to the design of DNN hardware accelerators. We have sketched an outline for a corresponding reconfigurable accelerator design, which utilizes this flexibility by adjusting the size of a fault protected hardware region. The actual realization of such an accelerator is an interesting field for future research.

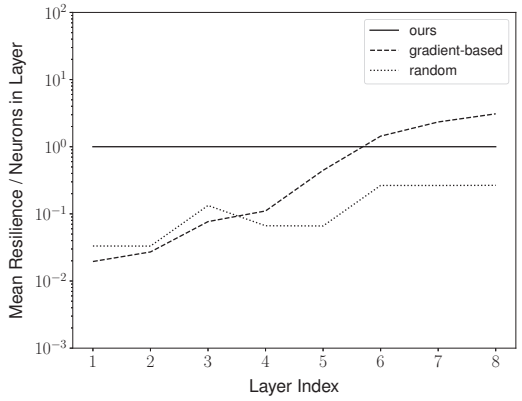
References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

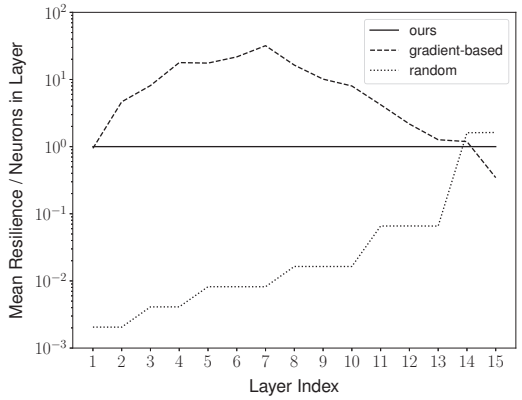


(a) Input Image (b) LRP-based (c) Gradient-based (d) Mean Difference

Fig. 5. Differences of the resilience assignments of the gradient-based method and the LRP-based method to the neurons of the first layer of the VGG-16 network averaged over the output feature maps.



(a) All-CNN (CIFAR-10)



(b) VGG-16 (ILSVRC)

Fig. 6. Layerwise mean neuron resilience assignment of the three methods.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra *et al.*, "Playing Atari with Deep Reinforcement Learning," *arXiv preprint*, 2013.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034.

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[5] A. Canziani, A. Paszke, and E. Culurciello, "An Analysis of Deep Neural Network Models for Practical Applications," *arXiv preprint*, 2016.

[6] ISO, "Road vehicles — Functional safety," 2011.

[7] J. Henkel, L. Bauer, N. Dutt, P. Gupta, S. Nassif, M. Shafique *et al.*, "Reliable on-chip systems in the nano-era," in *50th Annual Design Automation Conference*, 2013, pp. 695–704.

[8] R. Aitken, G. Fey, Z. T. Kalbarczyk, F. Reichenbach, and M. Sonza Reorda, "Reliability Analysis Reloaded: How Will We Survive?" in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2013, pp. 358–367.

[9] L. B. Gomez, F. Cappello, L. Carro, N. DeBardeleben, B. Fang, S. Gurumurthi *et al.*, "GPGPUs: How to combine high computational power with high reliability," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2014.

[10] S. Mittal, "A Survey of Techniques for Approximate Computing," *ACM Computing Surveys*, vol. 48, no. 4, pp. 1–33, 2016.

[11] E. B. Tchernev, R. G. Mulvaney, and D. S. Phatak, "Investigating the Fault Tolerance of Neural Networks," *Neural computation*, vol. 17, no. 7, pp. 1646–1664, 2005.

[12] J.-C. Vialatte and F. Leduc-Primeau, "A Study of Deep Learning Robustness Against Computation Failures," in *Ninth International Conference on Advanced Cognitive Technologies and Applications*, 2017.

[13] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller, "Explaining nonlinear classification decisions with deep Taylor decomposition," *Pattern Recognition*, vol. 65, pp. 211–222, 2017.

[14] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," *arXiv preprint*, 2013.

[15] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," *arXiv preprint*, 2014.

[16] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization," *arXiv preprint*, 2016.

[17] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling, "Visualizing Deep Neural Network Decisions: Prediction Difference Analysis," *arXiv preprint*, 2017.

[18] S. Venkataramani, A. Ranjan, K. Roy, and A. Raghunathan, "AxNN," in *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2014, pp. 27–32.

[19] Q. Zhang, T. Wang, Y. Tian, F. Yuan, and Q. Xu, "ApproxANN: An approximate computing framework for artificial neural network," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2015, pp. 701–706.

[20] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le *et al.*, "On rectified linear units for speech processing," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 3517–3521.

[21] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint*, 2014.

[22] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images."

[23] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations (ICLR)*, 2015.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[25] F. Chollet *et al.*, "Keras," 2015.

[26] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv preprint*, 2016.