A Study of Deep Learning Robustness Against Computation Failures

Jean-Charles Vialatte[1,2] and François Leduc-Primeau[1]

[1]Electronics Dept., IMT Atlantique, Brest, France

[2]Cityzen Data, Guipavas, France

emails: {jc.vialatte, francois.leduc-primeau}@imt-atlantique.fr

Abstract—**For many types of integrated circuits, accepting larger failure rates in computations can be used to improve energy efficiency. We study the performance of faulty implementations of certain deep neural networks based on pessimistic and optimistic models of the effect of hardware faults. After identifying the impact of hyperparameters such as the number of layers on robustness, we study the ability of the network to compensate for computational failures through an increase of the network size. We show that some networks can achieve equivalent performance under faulty implementations, and quantify the required increase in computational complexity.**

Index Terms—**Deep learning; quasi-synchronous circuits; energy-efficient computing.**

I. Introduction

Deep neural networks achieve excellent performance at various artificial intelligence tasks, such as speech recognition [1] and computer vision [2]. Clearly, the usefulness of these algorithms would be increased significantly if state-of-the-art accuracy could also be obtained when implemented on embedded systems operating with reduced energy budgets. In this context, the energy efficiency of the inference phase is the most important, since the learning phase can be performed offline.

To achieve the best energy efficiency, it is desirable to design specialized hardware for deep learning inference. However, whereas in the past the energy consumption of integrated circuits was decreasing steadily with each new integrated circuit technology, the energy improvements that can be expected from further shrinking of CMOS circuits is small [3]. A possible approach to continue improving the energy efficiency of CMOS circuits is to operate them in the near-threshold regime, which unfortunately drastically increases the amount of delay variations in the circuit, which can lead to functional failures [4]. There is therefore a conflict between the desire to obtain circuits that operate reliably and that are energy efficient. In other words, tolerating circuit faults without degrading performance translates into energy gains. Neural networks are interesting algorithms to study in this context, because their ability to process noisy data could also be useful to compensate for hardware failures. It is interesting to draw a parallel with another important class of algorithms that process noisy data: decoders of error-correction codes. For the case of low-density parity-check codes, it was indeed shown that a decoder can save energy by operating unreliably while preserving equivalent performance [5].

Of course, a perhaps more straightforward way to decrease the energy consumption is to reduce the computational complexity. For example, [6] proposes an approach to decrease the number of parameters in a deep convolutional neural network (CNN) while preserving prediction accuracy, and [7] proposes an approach to replace floating-point operations with much simpler binary operations while also preserving accuracy. The approach of this paper is the opposite. We consider increasing the number of parameters in the model to provide robustness that can then be traded for energy efficiency. The two aims are most likely complementary, and the situation is in fact similar in essence to the problem of data transmission, where it is in many practical cases asymptotically (in the size of the transmission) optimal to first compress the data to be transmitted, and then to add back some redundancy using an error-correction code.

In this preliminary study, we consider the ability of simple neural network models to increase their robustness to computation failures through an increase of the number of parameters. It was shown previously for an implementation of a multilayer perceptron (MLP) based on stochastic computing that it is possible to compensate for a reduced precision by increasing the size of the network [8]. The impact of hardware faults on CNNs is also considered in [9] using a different approach that consists in adding compensation mechanisms in hardware while keeping the same network size. The deviation models that we consider in this paper attempt to represent the case in which computation circuits are not protected by any compensation mechanism. We consider a pessimistic model and a more optimistic deviation model, but in both cases the error is only bounded by the finite codomain of the computations. We show that despite this, increasing the size of the network can sometimes compensate for faulty computations. The results provide an idea of the reduction in energy that must be obtained by a faulty circuit in order to reduce the overall energy consumption of the system.

The remainder of this paper is organized as follows. Section II provides a quick summary of the neural network models used in this paper, and Section III presents the methodology used for selecting hyperparameters of the models and for training. Section IV then describes the modeling of deviations, that is of the impact of circuit faults on the computations. Section V discusses the robustness of the inference based on simulation results. Finally, Section VI concludes the paper.

II. Background and Naming Conventions

A neural network is a neural representation of a function f that is a composition of *layer* functions f_i. Each layer function is usually composed of a linear operation followed by a non-linear one. A *dense layer* is such that its inputs and outputs are

vectors. Its linear part can be written as a matrix multiplication between an input x and a *weight matrix* W: $x \mapsto Wx$. The *number of neurons* of a dense layer refers to the number of rows of W. A n-*dimensional convolutional layer* is such that its inputs and outputs are tensors of rank n. Its linear part can be written as a n-dimensional convolution between an input x and a *weight tensor* W of rank $n+2$: $x \mapsto (\sum_p W_{pq} *_n x_p)_{\forall q}$, where p and q index slices at the last ranks and $*_n$ denotes the n-dimensional convolution. The tensor slices indexed by p and q are called *feature maps*. A pooling operation is often added to convolutional layers to scale them down.

An MLP [10] is composed only of dense layers. A CNN [11] is mainly composed of convolutional layers. For both network types, to perform supervised classification, a dense output layer with as many neurons as classes is usually added. Then, the weights are trained according to an optimization algorithm based on gradient descent.

III. NEURAL NETWORK MODELS

We consider two types of deep learning models and train them in the usual way, assuming reliable computations. To simplify model selection, we place some mild restrictions on the hyperparameter space, since we are more interested in the general robustness of the models, rather than in finding models with the very best accuracy. As described below, we restrict most layers to have the same number of neurons, and the same activation function. We also try only one optimisation algorithm, and consider only one type of weight initialization.

The first model type that we consider is an MLP network composed of L dense layers, each containing N neurons, that we denote as MLP–L–N. The activation function used in all the layers is chosen as the rectified linear unit (reLU) [12]. In fact, since a circuit implementation (and particularly, a fixed-point circuit implementation) can only represent values over a finite range, we take this into account in the training by using a clipped-reLU activation, which adds a saturation operation on positive outputs. We note that such an activation function has been used before in a different context, in order to avoid the problem of diverging gradients in the training of recurrent networks [13]. The use of the tanh activation was also considered, but reLU was observed to yield better accuracy.

The second model type is a CNN network composed of L convolutional $C \times C$ layers with $P \times P$ pooling of type *pool*, ultimately followed by a dense layer of 200 neurons [11]. This class is denoted as CNN–L–C–P–F–*pool*, where F is the number of feature maps used in each convolutional layer. Clipped-reLU activations are used throughout. The type of pooling *pool* can be either "max" pooling or no pooling.

For simplicity, we opted to train the networks on the task of digit classification using the MNIST dataset [14]. In addition to the layers defined above, each model is terminated by a dense "softmax" layer used for classification, and containing one neuron per class. Initialization was done as suggested in [15]. To prevent overfitting during training, a dropout [16]

of 25% and 50% of the neurons have been applied on convolutional and dense layers, respectively, except on the first layer. We used categorical crossentropy as the loss function and the "adadelta" optimizer [17]. The batch size was 128 and we trained for 15 epochs. The saturation value of the clipped-reLU activation is chosen as 1 as this provides a good balance between performance and implementation complexity. Note that as the range is increased, more bits must be used to maintain the same precision, leading to larger circuits.

IV. DEVIATION MODELS

We consider that all the computations performed in the inference phase are unreliable, either because they are performed by unreliable circuits, or because the matrix or tensor W is stored in an unreliable memory. We do, however, assume that the softmax operation performed at the end of the classification layer (i.e. the output layer) is performed reliably. We assume that each layer is affected by deviations independently, and that deviations occur independently for each scalar element of a layer output.

We are interested in circuits with a reasonably low amount of unreliability, and therefore it is useful to partition the outcome of a computation in terms of the occurrence or non-occurrence of a deviation event. We say that a deviation event occurs if the output of a circuit is different from the output that would be generated by a fully reliable circuit. The probability of a deviation event is denoted by p.

Obtaining a precise characterization of the output of a circuit when timing violations or other circuit faults are possible requires knowledge of the specific circuit architecture and of various implementation parameters. We will rely here on two simplified deviation models. We assume that the circuits operate on fixed-point values defined over a certain range, which is motivated by the fact that fixed-point computations are much simpler than floating-point ones, and reducing circuit complexity is the obvious first step in trying to improve energy efficiency. The first deviation model is a pessimistic model that assumes that when a deviation occurs, the output is sampled uniformly at random from the finite output domain of that circuit. We call this deviation model *conditionally uniform*. The second model is more optimistic, and assumes that the occurrence of a deviation can be detected. Therefore, when a deviation occurs, we replace the output by a "neutral" value, in this case 0. We call this deviation model the *erasure* model.

In a synchronous circuit, the deviations can be observed in the memory elements (registers) that separate the logic circuits. By changing the placement of these registers, we can in effect select the point where deviations will be taken into account. Note however that register placement cannot be arbitrary, as we also seek to have similar logic depth separating all registers. When considering the effect of the deviations on the inference, we noticed that robustness can be increased if deviations are sampled before the activation function of each layer. This is not surprising, since these activation functions act as denoising operations. All simulation results presented in this paper therefore consider that deviations are sampled before
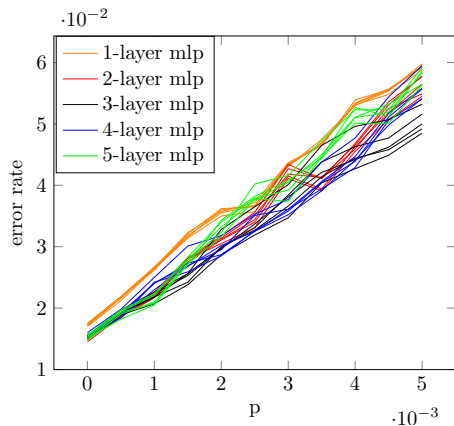
Fig. 1. Error rate of MLP under conditionally uniform deviations with prob. p. (Best viewed in color.)
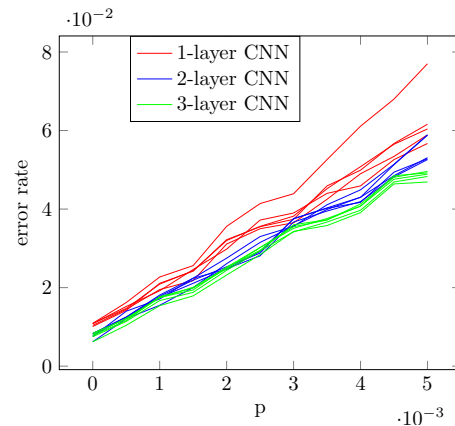

Fig. 2. Error rate of CNN under conditionally uniform deviations with prob. p. (Best viewed in color.)

the activation function. In the case of convolutional layers, the pooling operation is also performed after the sampling of deviations.

V. RESULTS

A. Effect of some hyperparameters on robustness

Robustness refers to the ability to maintain good classification accuracy in the presence of deviations. A model has better robustness if it achieves a lower classification error at a given p. We investigated the impact of several hyperparameters on the robustness of the inference. We evaluated the performance using the clipped-reLU and $\tanh$ activation functions, and found robustness to be similar in both cases. For CNN models, we also evaluated the impact of the choice of pooling function. In this case, we found that using no pooling, rather than max pooling, provided a slight improvement in robustness. Finally, we considered the impact of the number of layers L. The effect of L on classification error is shown in Figure 1 for an MLP-L-N network and in Figure 2 for a CNN-L-C-P-F-pool for the case where the inference is affected by conditionally uniform deviations. For each value of L, we select the 5 best models based on their deviation-free performance to show the variability of the model optimization. In the case of MLP models, the number of layers does not have a clear impact on robustness. However, while the 2-layer models minimize error under reliable inference, we see that models with more layers sometimes minimize error when the deviation probability increases. However, in the case of CNN models, we can see that increasing the number of layers usually improves robustness.

B. Fault tolerance

Since neural networks are designed to reject noise in their input, we might expect them to also be able to reject the "noise" introduced by faulty computations. We investigate this ability for networks trained with standard procedures, in order to provide a baseline for future targeted approaches.

We first choose an error rate target that we want the network to achieve. We then consider various deviation probability

values, and for each, look for a model with the smallest number of parameters that can achieve or outperform the performance target under the deviation constraint. For MLP networks, the results are shown in Figure 3 for the case of conditionally uniform deviations, and in Figure 4 for the case of erasure deviations. Similarly, Figures 5 and 6 show the results for CNN networks, respectively for conditionally uniform and erasure deviations.

We consider deviation probabilities on the order of 10^{-3}, which are already considered quite large for digital circuit design. We can see that in all cases, there are indeed many performance targets at which it is possible to compensate computation failures by increasing the number of parameters in the model. However, if one wishes to obtain the best performance achievable by deviation-free inference under conditionally uniform deviations, there is no parameter increase that can compensate for $p \geq 10^{-3}$ within the model sizes that were considered. On the other hand, all performance targets can be achieved under erasure deviations.

Finally, we can note that the CNN models are much more robust than MLP models. For instance, under conditionally uniform deviations and with an error rate target of 0.024, an MLP requires a 318% increase in the number of parameters to tolerate a deviation probability $p = 10^{-3}$, while a CNN can achieve an error rate target of 0.023 at $p = 10^{-3}$ with only a 0.2% increase in the number of parameters.

VI. CONCLUSION

In this paper, we have considered the effect of faulty computations on the performance of MLP and CNN inference, using a pessimistic and an optimistic deviation model. We showed that using standard training procedures, it is in many cases possible to find models that will compensate for computation failures, when the deviation probability is on the order of 10^{-3} and at the cost of an increase in the number of parameters. This increase is generally small for CNNs or when modeling deviations as erasures, but quite large for the case of MLP models under conditionally uniform deviations. These
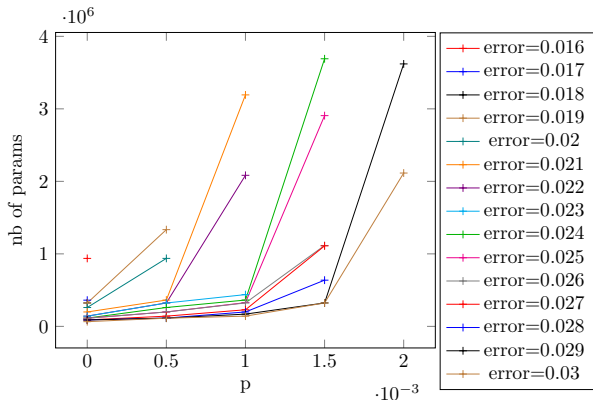
Fig. 3. Number of model parameters needed to achieve error rate target using MLP under conditionally uniform deviations.
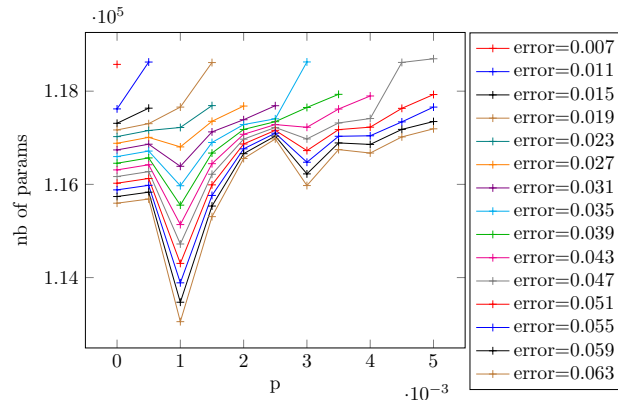

Fig. 5. Number of model parameters needed to achieve error rate target using CNN under conditionally uniform deviations.
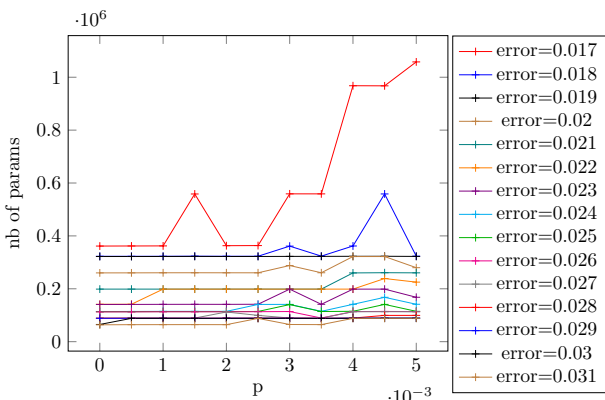

Fig. 4. Number of model parameters needed to achieve error rate target using MLP under erasure deviations.
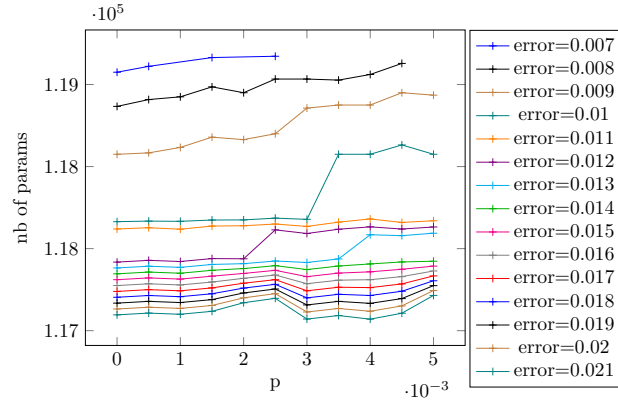

Fig. 6. Number of model parameters needed to achieve error rate target using CNN under erasure deviations.

results provide a baseline for future work seeking to identify systematic ways of designing robust deep neural networks.

REFERENCES

[1] D. Amodei *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," *CoRR*, vol. abs/1512.02595, 2015. [Online]. Available: http://arxiv.org/abs/1512.02595

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[3] R. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-threshold computing: Reclaiming Moore's law through energy efficient integrated circuits," *Proc. of the IEEE*, vol. 98, no. 2, pp. 253–266, Feb. 2010.

[4] M. Alioto, "Ultra-low power VLSI circuit design demystified and explained: A tutorial," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 59, no. 1, pp. 3–29, 2012.

[5] F. Leduc-Primeau, F. R. Kschischang, and W. J. Gross, "Modeling and energy optimization of LDPC decoder circuits with timing violations," *CoRR*, vol. abs/1503.03880, 2015. [Online]. Available: http://arxiv.org/abs/1503.03880

[6] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1mb model size," *CoRR*, vol. abs/1602.07360, 2016. [Online]. Available: http://arxiv.org/abs/1602.07360

[7] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," *CoRR*, vol. abs/1603.05279, 2016. [Online]. Available: http://arxiv.org/abs/1603.05279

[8] A. Ardakani, F. Leduc-Primeau, N. Onizawa, T. Hanyu, and W. J. Gross, "VLSI implementation of deep neural network using integral stochastic computing," *IEEE Trans. on VLSI Systems*, 2017, to appear.

[9] Y. Lin, S. Zhang, and N. R. Shanbhag, "Variation-tolerant architectures for convolutional neural networks in the near threshold voltage regime," in *2016 IEEE International Workshop on Signal Processing Systems (SiPS)*, Oct 2016, pp. 17–22.

[10] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[12] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[13] A. Y. Hannun *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *CoRR*, vol. abs/1412.5567, 2014. [Online]. Available: http://arxiv.org/abs/1412.5567

[14] Y. LeCun, C. Cortes, and C. J. Burges, "The MNIST database of handwritten digits," 1998.

[15] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[17] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012. [Online]. Available: http://arxiv.org/abs/1212.5701