ISSN 1060-992X, Optical Memory and Neural Networks, 2019, Vol. 28, No. 2, pp. 82–88. © Allerton Press, Inc., 2019.

Study of Fault Tolerance Methods for Hardware Implementations of Convolutional Neural Networks

R. A. Solovyev[a], *, A. L. Stempkovsky[a], and D. V. Telpukhov[a]

[a] *Institute for Design Problems in Microelectronics, Russian Academy of Sciences, Moscow, 124681 Russia*

**e-mail: turbo@ippm.ru*

Received March 25, 2019; revised March 25, 2019; accepted April 5, 2019

Abstract—The paper concentrates on methods of fault protection of neural networks implemented as hardware operating in fixed-point mode. We have explored possible variants of error occurrence, as well as ways to eliminate them. For this purpose, networks of identical architecture based on VGG model have been studied. VGG SIMPLE neural network that has been chosen for experiments is a simplified version (with smaller number of layers) of well-known networks VGG16 and VGG19. To eliminate the effect of failures on network accuracy, we have proposed a method of training neural networks with additional dropout layers. Such approach removes extra dependencies for neighboring perceptrons. We have also investigated method of network architecture complication to reduce probability of misclassification because of failures in neurons. Based on results of the experiments, we see that adding dropout layers reduces the effect of failures on classification ability of error-prone neural networks, while classification accuracy remains the same as of the reference networks.

Keywords: convolutional neural networks (CNN), dropout, fault tolerance

DOI: 10.3103/S1060992X19020103

INTRODUCTION

Currently, convolutional neural networks (CNN) are widely used in microelectronic computing devices for solving problems related to computer vision. Neural networks already proved high efficiency in this field, and therefore problems of efficient hardware implementation [1] and neurochip design are extremely acute nowadays. Generally, the efficiency is understood as increasing requirements related to performance, area and power consumption. However, wide use of neural networks in the field of control systems in transport and other critical applications leads to actualization of new problems associated with ensuring advanced reliability of neurochip operation.

When microelectronic devices are exposed to destabilizing factors such as radiation, high temperatures, etc., faults and failures occur, which can lead to catastrophic consequences because of incorrect operation of the device. One of the promising approaches to enhance reliability of microelectronic systems is the radiation-resistant design paradigm – the so-called RHBD technology (Radiation Hardening by Design), within which a large set of various methods are used, such as multiple redundancy [2], noise-resistant coding [3], as well as other architectural methods at circuit and topology levels [4]. The named approaches are applicable to wide class of devices. Hardware implementations of neural networks are among them.

At the same time, specific features of neural networks operation open prospects to nonstandard approaches of reliability improvement. In particular, in this work we propose usage of dropout layers during neural networks training process, thus improving reliability characteristics of the devices. The research has shown that this technique is quite effective and can be used in addition to the existing methods of fault tolerance improvement.

1. STRUCTURE OF CONVOLUTIONAL NEURAL NETWORKS (CNN)

Convolutional neural networks are a type of neural networks designed for image recognition and detecting objects in pictures. They were proposed in 1988 by Yan Lecun [6]. Architecture of such networks is built by analogy with human visual cortex. CNN was named because of sequential execution of convo-

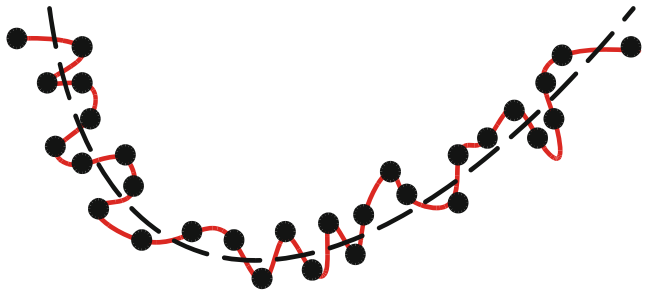

Fig. 1. Demonstration of regularized model (black line) and overfitting model (red line).

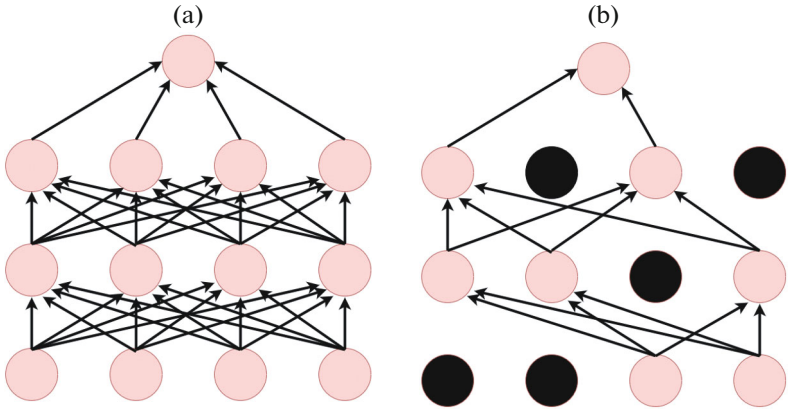

Fig. 2. Graphical representation of dropout layer operation a) standard network b) network with dropout.

lution of two-dimensional matrices. Error back propagation is used as training algorithm for such networks.

CNN structure is multilayer and unidirectional; usually it consists of the following layers:

(1) Input (input data)—original images are fed to it.

(2) Conv2D (two-dimensional convolution layer)—convolution is applied to input data. In most cases, convolution core size is 3 × 3.

(3) MaxPooling2D (pooling layer) performs down-sampling in spatial dimensions (width and height). For example, 28 × 28 image is converted to 14 × 14 image. For each 2 × 2 segment maximum value is selected; in doing so, the other values are left out. Sometimes this layer is omitted in modern networks and convolutions with step equal to 2 are used.

(4) Dense (fully connected layer)—a layer in which each neuron is connected with all neurons from the previous layer. It requires large amount of calculations and storage of large number of weights. Usually it is one of the last layers of the neural network and serves for classifying result.

(5) Activation—these layers add nonlinearity. Most often Relu, Sigmoid, Tanh and their variations are used.

(6) Additional layers: BatchNormalization [7], GlobalAveragePooling and so on.

2. DROPOUT LAYER AND ITS CONVENTIONAL APPLICATION

There are many parameters when training neural networks, such as input data order, activation function, target values of neuron outputs, initialization of weights, learning rate and so on. Also, developers often meet a number of problems, such as low training speed, large overhead costs for storing weights, network overfitting on the given test data. Different methods are used to solve these problems, and one particular solution is dropout layers. Dropout is a method for regularizing artificial neural networks, which is primarily intended to prevent CNN overfitting.

When neural network is trained on a dataset containing few elements, overfitting is a frequent problem. The neural network ceases to pick out any characteristic features and just remembers the training sample

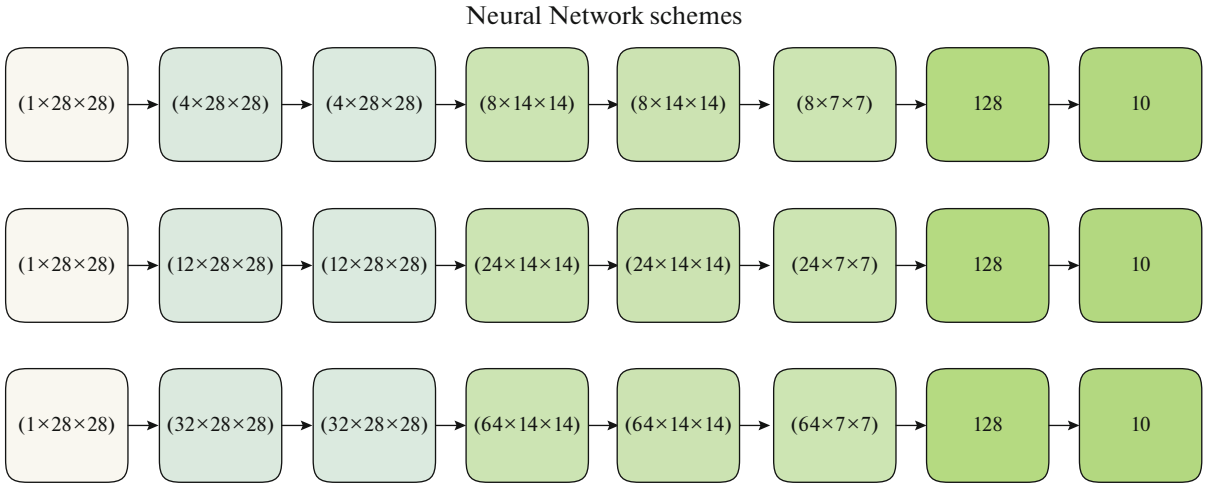

Fig. 3. Schematic structures of trained networks of VGG SIMPLE family.

as a certain pattern. Overfitting is a process in which network model makes qualitative prediction on the training samples only; when third-party examples that did not participate in the training are submitted, we get much worse result. An example of such a situation can be easily seen in Fig. 1. Each neuron of the network tends to select only those features that help solve the problem (predict regression model, classify object, etc.). Dropout layer helps us to solve the overfitting problem [8] by "dropping out" a random subset of neurons at each training iteration; this prevents complex joint adaptation of weights to training data (Fig. 2).

3. FORMULATION OF THE BASIC ASSUMPTION

As demonstrated in the previous section, dropout makes it possible to reduce the dependence of each individual neuron of the layer from values of the previous layer neurons. Therefore, it was suggested that a neural network trained in this way, in case of a fault in one of the weights, is more likely to mask the error by the effect of subsequent layers of the network.

Dropout is usually applied only at the last "classification" layers of CNN, however, it is possible to use these layers practically after each internal layer, which will reduce the influence of each individual neuron of the intermediate layers on the overall result of calculations. Experiments have shown that adding dropout layers in the middle of the network has almost no effect on the classification accuracy and only slightly reduces the training speed.

The study assumes that weights of neural network are stored in memory all the time of its functioning, and they are mostly subject to failures and tend to accumulate errors, rather than the results of intermediate calculations. Thus, the proposed method is applicable to the problem of ensuring resilience to inversions in memory cells—single event upset (SEU), as opposed to methods aimed at ensuring fault tolerance of combinational circuits.

4. USING DROPOUT LAYER FOR IMPROVING RELIABILITY OF NEURAL NETWORK OPERATION

As part of the research, computational experiments were conducted: single, double, and five-fold errors were injected in the weights of neural network, thereby simulating failures in memory cells. For testing, VGG Simple network [9] based on the VGG architecture [10] was chosen, which is a simplified analogue of VGG16 and VGG19 networks. The neural network VGG Simple was chosen for reasons of simplicity and high training speed as it has half the perceptron layers and less filters on each layer. For the experiment, several neural networks built on this architecture were trained. They differ in number of filters on each of the layers. Structures of the trained networks are presented below in Fig. 3. Python code for the network is shown in Fig. 4. The networks were trained on MNIST dataset for recognizing handwritten numbers [11]. A set of 50000 images was used for training. After that, the weights of the networks were kept as a reference.

```
def VGG Simple():
    model – Sequential()
    model.add(Conv2D(4, (3, 3), padding = 'same', activation = 'relu', input_shape = input_shape))
    model.add(Conv2D(4, (3, 3), activation = 'relu', padding = 'same'))
    model.add(MaxPooling2D(pool_size = (2, 2)))
    model.add(Conv2D(8, (3, 3), padding = 'same', activation = 'relu'))
    model.add(Conv2D(8, (3, 3), activation = 'relu'))
    model.add(MaxPooling2D (pool_size = (2, 2)))
    model.add(Flatten())
    model.add(Dense(128, activation = 'relu'))
    model.add(Dense(nb_classes, activation = 'softmax'))
return model
```

Fig. 4. Python code (Keras module) describing CNN VGG Simple.

```
def VGG_Simple_drop():
    model – Sequential()
    model.add(Conv2D(4, (3, 3), padding = 'same', activation = 'relu', input_shape = input_shape))
    model.add(Dropout(0.25))
    model.add(Conv2D(4, (3, 3), activation = 'relu', padding = 'same'))
    model.add(Dropout(0.25))
    model.add(MaxPooling2D(pool_size = (2, 2)))
    • model.add(Conv2D(8, (3, 3), padding = 'same', activation = 'relu'))
        model.add(Dropout(0.25))
        model.add(Conv2D(8, (3, 3), activation = 'relu'))
        model.add(Dropout(0.25))
        model.add(MaxPooling2D (pool_size = (2, 2)))
model.add(Flatten())
    model.add(Dense(128, activation = 'relu'))
    model.add(Dropout(0.5))
    model.add(Dense(nb_classes, activation = 'softmax'))
return model
```

Fig. 5. Python code for CNN VGG SIMPLE with dropout blocks added after each layer.

Table 1. Percentage of results matching the reference after single error injection into convolutional layer (Conv)

Conv layer number	Filter number	No dropout	Dropout (0.5)	Dropout (0.25)	Dropout (0.1)
1	4	**93.17**	93.41	97.66	98.90
1	12	99.85	99.75	99.80	99.89
1	32	99.97	99.94	99.94	99.93
2	4	**98.03**	98.69	99.08	98.57
2	12	99.92	99.94	99.94	99.94
2	32	99.99	99.99	99.99	99.99
3	8	**95.60**	97.74	98.38	98.88
3	24	99.91	99.92	99.89	99.90
3	64	99.98	99.98	99.98	99.97
4	8	**96.62**	99.71	99.62	99.54
4	24	99.85	99.95	99.93	99.88
4	64	99.98	99.99	99.99	99.97

Then the same networks were modified by adding intermediate dropout layers. Python code for such network is shown in Fig. 5. These new networks were re-trained on the same dataset. As a result of the training, high classification accuracy was achieved (about 97%) for all these networks. Networks using dropout had the same accuracy as without dropout, within the operational margin; however, they required a little more training time.

The experiment consisted in injecting errors of different multiplicity into neural network weights; after that test images were processed. The results were compared with the reference, that is, the same network without errors.

5. EXPERIMENTAL RESULTS

In experiments, pre-trained neural networks were alternately used, total of 12 CNN for testing, which differed in size of convolutional blocks (sizes from the list [4, 12, 32]) and values of dropout: 0 (without dropout), 0.1, 0.25 and 0.5. Error was injected randomly into one of the network layers tested in the experiment, after which test data in amount of 10000 images was passed through the network. The result of measurements was compared with the reference, that is, the CNN without injected errors. Further, percentage of the number of images, for which classification result matched the reference, was considered. The value of 100% indicated that the injected errors did not affect the result of neural network operation. Maximum size of error injected into the weights was:

$$\text{error} = 2|w_{\max}|.$$

This value was chosen in consideration that the weight in the memory is stored as binary notation and when changing random bits, for each individually selected weight, the changed weight cannot differ from it more than twice the maximum possible weight for a given layer.

Results for Single Error Injection into Convolution Layers

From Table 1 we can see that after single error injection the result of the network with small number of neurons and without dropout degraded to 93% correct answers. At the same time, the use of dropout method with the value of 0.1 improved this result to almost 99%. From the same table we see that neural networks that have greater number of neurons almost do not respond to a single error in the layers.

Results for Double Error Injection into Convolution Layers

A double failure has greater effect on the final result of the networks (see Table 2). So, if a failure was brought into the third layer of the convolution, the percentage of correct answers compared with the network without failures dropped to 83% (for networks with small number of neurons). Properly selected

Table 2. Percentage of results matching the reference after double error injection into convolutional layer (Conv)

Conv layer number	Filter number	No dropout	Dropout (0.5)	Dropout (0.25)	Dropout (0.1)
1	**4**	**90.87**	88.86	96.29	98.40
1	12	99.79	99.62	99.74	99.84
1	32	99.95	99.92	99.90	99.90
2	**4**	**88.81**	92.51	97.24	97.70
2	12	99.88	99.92	99.89	99.94
2	32	99.99	99.99	99.97	99.98
3	**8**	**83.30**	88.88	97.81	98.06
3	24	99.77	99.80	99.84	99.73
3	64	99.96	99.98	99.95	99.96
4	**8**	**88.10**	98.12	99.35	98.52
4	24	99.72	99.90	99.90	99.85
4	64	99.96	99.98	99.96	99.97

Table 3. Percentage of results matching the reference after five-fold error injection into convolutional layer (Conv) for network with small number of filters (4–8)

Conv layer number	Filter number	No dropout	Dropout (0.5)	Dropout (0.25)	Dropout (0.1)
1	4	88.65	87.57	94.46	96.83
2	4	60.13	73.24	80.57	76.45
3	8	57.58	65.55	75.59	60.30
4	8	41.48	86.07	82.58	62.03

dropout value helped to raise this result to 97–98%. At the same time, the use of dropout method with the value 0.1 corrected this result.

Results for Five-fold Error Injection into Convolution Layers

Five-fold error is critical for networks with small number of neurons (see Table 3). The proposed method cannot improve the result significantly, although we note that, when using dropout, the result is much better. Large neural networks throughout all experiments have expectedly shown better error resistance.

Results for Error Injection into Fully Connected Layers

From Table 4 we can see that fully connected layers are generally more error tolerant than convolutional layers. Most likely, this is due to the fact that each neuron is connected to each neuron of the previous layer, in contrast to the convolutional layers, where only a part of the previous layer neurons are involved. In general, there are more neurons in the previous fully connected layer than in convolutional layers, and the error in one or several of them has less effect on the result of the next layer due to the larger number of weights in formula for calculating next layer values. Here, too, the same trend is observed – the presence of dropout after fully connected layers increases fault-tolerance of neural networks.

6. CONCLUSIONS

In general, we can draw the following conclusions based on the results of the study:

(1) Adding dropout layers helps to reduce the effect of errors on the result of neural networks classification in presence of destabilizing factors.

(2) Networks with greater number of neurons are more resilient to error injection.

(3) As discussed in this paper, even relatively small neural networks easily tolerate multiple errors, usually without significant drop in classification accuracy. We can expect that large industrial neural networks (VGG16, VGG19) can withstand errors of higher multiplicity.

(4) Fully connected layers are more resistant to errors than convolutional ones, as expected, due to the greater number of weight connections between perceptrons.

Table 4. Percentage of results matching the reference after error injection into fully connected layers (Dense)

Dense layer number	Size	No dropout	Dropout (0.5)	Dropout (0.25)	Dropout (0.1)
Single error					
1	Small	99.964	99.974	99.985	99.963
1	Medium	99.976	99.997	99.993	99.990
1	Large	99.993	99.998	99.997	99.996
2	Small	99.348	99.955	99.923	99.893
2	Medium	99.754	99.951	99.944	99.908
2	Large	99.912	99.971	99.957	99.952
Double error					
1	Small	99.797	99.930	99.972	99.938
1	Medium	99.966	99.990	99.985	99.979
1	Large	99.989	99.997	99.993	99.990
2	Small	99.030	99.874	99.826	99.669
2	Medium	99.418	99.898	99.855	99.822
2	Large	99.759	99.946	99.918	99.875
Five-fold error					
1	Small	98.950	99.643	99.690	99.160
2	Small	96.837	99.527	99.568	99.445

CONFLICT OF INTEREST

The authors declare that they have no conflict of interest.

REFERENCES

1. Matyushkin, I.V. and Solovyev, R.A., A model of adaptive neuron and its hardware implementation on PLD, *Elektron. Tekh., Ser. 3: Mikroelektron.,* 2017, no. 3, pp. 53–61.
2. Telpukhov, D.V., Rukhlov, V.S., Ivanova, G.A., Ryzhova, D.I., Nadolenko, V.V., and Demeneva, A.I., The study of options for partial redundancy in the design of fail-safe logic PLD blocks, *Inzh. Vestn. Dona,* 2018, no. 1, p. 47.
3. Gavrilov, S.V., Gurov, S.I., Zhukova, T.D., Rukhlov, V.S., Ryzhova, D.I., Telpukhov, D.V. Methods to improve the failure resistance of combinational ICs based on redundant coding, in *Prikladnaya matematika i informatika trudy fakul'teta VMK MGU imeni M.V. Lomonosova* (Applied Mathematics and Computer Science, Proceedings of the Department of Computational Mathematics and Cybernetics of the Lomonosov Moscow State University), Moscow, 2016, pp. 93–102.
4. Adamov, D.Yu., Adamov, Yu.F., and Balaka, E.S., Analysis of the stability of CMOS of differential amplifiers to the effects of accumulated dose of ionizing radiation, *Izv. Vyssh. Uchebn. Zaved., Elektron.,* 2016, vol. 21, no. 2, pp. 145–151.
5. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R.R., Improving neural networks by preventing co-adaptation of feature detectors, 2012, *arXiv preprint arXiv:1207.0580.*
6. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W. and Jackel, L.D., Backpropagation applied to handwritten zip code recognition, *Neural Comput.,* 1989, vol. 1, no. 4, pp. 541–551.
7. Ioffe, S. and Szegedy, C., Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. *arXiv preprint arXiv:1502.03167.*
8. Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., and Fergus, R., Regularization of neural networks using DropConnect, *International Conference on Machine Learning,* 2013, pp. 1058–1066.
9. Solovyev, R.A., Kustov, A.G., Rukhlov, V.S., Shelokov, A.N., and Puzyrkov, D.V., Hardware implementation of the convolutional neural network on PLD based on fixed-point calculations, *Izv. Yuzhn. Fed. Univ., Tekh. Nauki,* 2017, no. 7, pp. 186–197.
10. Simonyan, K. and Zisserman, A., Very deep convolutional networks for large-scale image recognition, 2014, *arXiv preprint arXiv:1409.1556.*
11. LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P., Gradient-based learning applied to document recognition, *Proceedings of the IEEE,* 1998, vol. 86, no. 11, pp. 2278–2324.