Applied Mathematics and Computation 267 (2015) 119–138

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

An efficient initialization mechanism of neurons for Winner Takes All Neural Network implemented in the CMOS technology

Tomasz Talaśka [a,*], Marta Kolasa [a], Rafał Długosz [a,b], Pierre-André Farine [b]

[a] UTP University of Science and Technology, Faculty of Telecommunications, Computer Science and Electrical Engineering, ul. Kaliskiego 7, 85-796 Bydgoszcz, Poland
[b] Institute of Microtechnology of Swiss Federal Institute of Technology in Lausanne, Rue de la Maladière 71B, CH-2002 Neuchâtel, Switzerland

ARTICLE INFO

Keywords:
Winner Takes All Neural Network
Initialization mechanism
Convex Combination Method
CMOS implementation
Analog circuits
Low energy consumption

ABSTRACT

The paper presents a new initialization mechanism based on a Convex Combination Method (CCM) for Kohonen self-organizing Neural Networks (NNs) realized in the CMOS technology. A proper selection of initial values of the neuron weights exhibits a strong impact on the quality of the overall learning process. Unfortunately, in case of real input data, e.g. biomedical data, proper initialization is not easy to perform, as an exact data distribution is usually unknown. Bad initialization causes that even 70%–80% of neurons remain inactive, which increases the quantization error and thus limits the classification abilities of the NN. The proposed initialization algorithm has a couple of important advantages. Firstly, it does not require a knowledge of data distribution in the input data space. Secondly, there is no necessity for an initial polarization of the neuron weights before starting the learning process. This feature is very convenient in case of transistor level realizations. In this case the programming lines, which in other approaches occupy a large chip area, are not required. We proposed a modification of the original CCM algorithm. A new parameter which in the proposed analog CMOS realization is represented by an external current, allows to fit the behavior of the mechanism to NNs containing different numbers of neurons. The investigations show that the modified CCM operates properly for the NN containing even 250 neurons. A single CCM block realized in the CMOS 180 nm technology occupies an area of 300 μm^2 and dissipates an average power of 20 μW and at data rate of up to 20 MHz.

1. Introduction

Artificial neural networks (ANNs) are universal and efficient tools used in solving various problems including identification, classification, data compression and others. Many examples of successful application of ANNs in such areas as electronics, electrical engineering, mechanical engineering, economics, health care and others can be found in literature. In the artificial intelligence (AI) research area the main effort is usually focused on the development of new architectures of Neural Networks (NN) and more efficient learning algorithms. The aim of these investigations is to improve the quality, as well as the speed of the learning process. In majority of reported cases ANNs are realized in software, which is due to large

* Corresponding author.
E-mail addresses: talaska@utp.edu.pl (T. Talaśka), markol@utp.edu.pl (M. Kolasa), rafal.dlugosz@gmail.com (R. Długosz), pierre-andre.farine@epfl.ch (P.-A. Farine).

http://dx.doi.org/10.1016/j.amc.2015.04.123
0096-3003/© 2015 Elsevier Inc. All rights reserved.

flexibility of such implementations. If the NN does not work properly after the implementation, then it can be easily and quickly reprogrammed.

In our work we focus on ANNs realized as Application Specific Integrated Circuits (ASICs). Designing such systems is much more difficult than those realized in software. However, in this case the Figure-of-Merit defined, for example, as data rate over the power dissipation can be even several orders of magnitude larger than in counterpart software realizations [1–4]. The achieved improvements of the computation power are mostly due to parallel and asynchronous data processing being realized in such NNs. Low power consumption, as well as low chip area result from the possibility of an accurate matching of the internal structure of the circuit to realized functions. This is possible mainly in case of ASICs realized in the 'full-custom' style. NNs being trained in an unsupervised manner, such as Winner Takes All (WTA) and Winner Takes Most (WTM) self-organizing maps (SOMs) are of our particular interest [4–7]. The rationale behind selection of these algorithms is their high efficiency on the one hand, and simplicity, on the other one. Learning algorithms in this case require only basic arithmetical operations and thus can be easily implemented at the transistor level.

A typically long design process of such systems is one of the essential reasons why hardware implementations are still quite rarely considered in practice in comparison with existing software implementations. While there is some rationale behind such arguments, the recent developments of electronic circuits, in particular target applications where ultra low energy consumption is of the paramount relevance, justify new investigations in the area of hardware realized ANNs. For example, in wireless sensor networks (WSN) a substantial effort has been put to develop devices which are self-sufficient in energy supply [8–10]. This is a class of circuits in which hardware implemented ANNs may find a wide array of applications.

The basic question usually asked before starting designing a system of this type is whether to use analog, digital or mixed technique. Analog circuits usually suffer from various phenomena such as a transistor mismatch, a leakage in analog memory cells, off-sets in comparators, etc, however offer a compact structure of circuits that sometimes perform even very complex functions. Digital circuits, on the other hand, are much more complex, yet simultaneously more robust against the undesirable phenomena listed above. Depending on the type of the realized block and the target application different techniques are used.

The work presented in this paper is a continuation of our former project, in which we developed from scratch a fully analog NN [4,5]. This is one of the reasons why the presented circuit is realized in analog technique. However, we do not limit the presented work to implementation aspects only. In fact, we propose a new algorithm that can be classified either as an efficient initialization mechanism but also, to some extent, as a modification of the WTA learning algorithm. The presented results can be also used in digital ASICs, as well as in pure software systems.

1.1. Problems associated with the initialization of neuron weights

The efficiency of training of ANNs depends on many circumstances. One of them is a proper polarization of neurons weights before starting the learning phase. The weights can be viewed as coordinates that determine the location of particular neurons in an m-dimensional input data space, where m is the number of inputs of the NN. A proper distribution of neurons in this space before proceeding with the learning phase has a direct impact on the convergence speed of the learning process, as well as on the final training results [11–14]. In this work we focus on NNs trained in the unsupervised manner. In such networks properly selected initial values of the weights have a strong impact on the number of the so-called dead neurons [4]. Such neurons take part in the competition, but never win and therefore never become representatives of any data class. This increases the quantization error of the NN [4,15,16] as discussed in Section 2.2.

Various initialization methods can be found in literature, but one universal method suitable for all learning algorithms does not exist. The initialization process of neuron weights should reflect data distribution in a training dataset. The problem is that such distribution is usually not known in advance. For this reason, the initial values of the weights are usually determined either empirically or randomly. More sophisticated methods also exist, but they are usually too complex to be easily implemented at the transistor level [17,18]. The problem of the initialization can be considered from two different points of view. Since NNs are usually implemented in software, the problem of the computational complexity of a given initialization algorithm is of less relevance in this case. Taking into account the computational capabilities of computers used today, the most important criterion is the efficiency of a given algorithm. In case of NNs realized as ASICs the situation is opposite. In this case we have to face with various hardware inaccuracies and limitations that have an influence on the initialization process.

Various initialization methods have been proposed over the past twenty years [13,19–21,23,24]. In a very common and simple approach a random and uniform distribution of the weights over the input data space is being applied [19–21]. This approach is very fast, which is one of its main advantages. However it is not always effective, as it does not reflect a distribution of data over the input data space, which is usually not known. As regards to ANNs realized in hardware working in parallel, in which each neuron is a separate circuit, this method generates several additional problems. One of them is especially visible in large NNs, in which a net of programming lines connected to particular neuron weights makes the layout very complex. Due to usually limited number of pins in the chip, the weights have to be programmed sequentially, which requires an additional circuitry responsible for addressing particular memory cells. The two described problems can be classified as implementation issues. There are also complications that occur during the operation of the NN. Potentials on the programming lines usually differ from the values of the weights stored as voltages in corresponding memory cells.