CAPABILITIES OF HARDWARE SUPPORT IN CREATION OF ARTIFICIAL NEURAL NETWORKS

ARTUR BARTOSZEWSKI[1]

Abstract

The paper is devoted to capabilities of hardware support in creation of artificial neural networks. This paper describes possibility of building hardware neuro-processors, possibility of use SIMD instructions implemented in modern processors and the ability to use unified shader graphics card. The author shows an example of a simple implementation of a single neuron with the use of GPU shader.

Key words: Artificial neural network, SIMD, GPU, shader

1. Introduction

Neural networks that have been developed long time ago, are still a promising technology. Their idea is based on knowledge of the functions of the human brain. Neural networks can be used to solve problems for which it is not possible to create an exact algorithm. Neural networks can be used in

- identifying and classifying patterns
- time series prediction,
- analysis of statistical data
- audio and video processing
- automatic control of processes

There are many practical examples of tasks performed by artificial neural networks. These tasks include: diagnosis of electronic systems; psychiatric test; stock market predictions; sales forecasting; exploration for oil; interpretation of biological tests; price forecast; analysis of medical research; machines maintenance planning; planning progress in learning; analysis of production problems; optimization of business; spectral analysis; optimization of waste disposal; selection of raw materials; selection purposes in forensic investigations; selection of employees; control of industrial processes [1, p. 13-14] and many others.

Artificial neural networks can be constructed as hardware. With the use of special neural-processors. Much more often, neural networks are implemented as software running on the classic, (based on von Neumann architecture) computers. Many of these tasks are performed by highly complex and powerful neural networks. Implementing them requires a computer with high computing power. Often these are computers (cluster computers) designed specifically for this task.

[1] Technical Uniwersity of Radom, Poland

The average computer user does not have access to such equipment (we don't discuss here the opportunities offered by cloud computing). Previous attempts to widespread use of neural networks on the PC did not produce the expected results. A good example is OCR (Optical Character Recognition). Developers create self-learning programs using neural networks, but simple networks do not meet the demands of users. Sometime later, developers resigned from the their use in favor of other solutions (often heuristics).

The widespread use of neural networks requires the optimum use of existing hardware solutions. The use of technologies implemented in the processor IA-64 family (modern successor to the x86 architecture), allows you to create more complex neural networks. Even more powerful is to use the GPU (Graphics Processing Unit) to calculate the network.

To understand how to use the capabilities here described it is needed to know the specificity of artificial neural networks.

2. The specificity of neural networks

The neural network consists of many (sometimes it is a very large number) of simple elements that are capable of parallel processing. Joint and simultaneous action of these elements enables the processing of input signals to output signals (results).

Neuron can be regarded as a simple processor. This may be described as follows:

- neuron receives multiple input signals;
- neuron has one output signal (response);
- to each input neuron is assigned a parameter called weight;
- each input signal is multiplied by the weight of this input;
- after multiplying, input signals are summed, this sum is called the net value;
- in some types of neural networks to the sum signal is added an additional component (independent of the input signal) is called bias.

In many types of networks the sum of the input signal multiplied by the weight of the inputs is treated as a result (output neuron). In the MLP type networks (Multilayer Perceptron) output signal is calculated using the nonlinear function that describes the dependence of the output neuron of the net value. Relation between input signals and output signals of the neuron is called the transfer function. For a simple, linear neuron transfer function is as follows:

$$y = \sum_{i=1}^{n} w_i x_i \tag{1}$$

where: *n - number of inputs,*
w - the weight of input
x - input signal.

Transfer function of the nonlinear neuron (2) is more complicated.

$$\begin{aligned} y &= \varphi(e) \\ e &= \sum_{i=1}^{n} \left(w_i x_i + \phi \right) \end{aligned} \tag{2}$$

where: *n - number of inputs,*
w - the weight of input
x - input signal,
φ *- activation function (nonlinear)*
ϕ *- bias (fixed component).*

Diagram of a single neuron is shown in Figure 1.

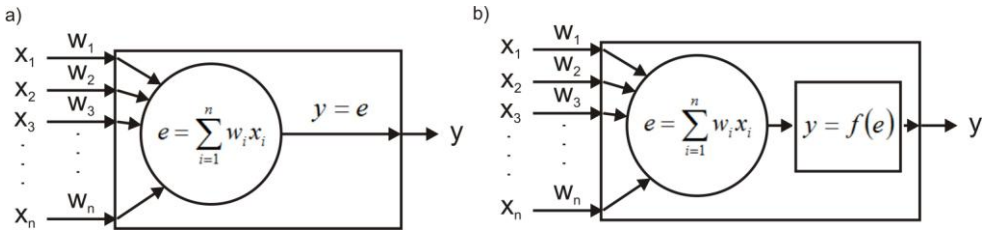

Fig. 1. Diagram of the neuron a) linear b) nonlinear

The main difference between linear neuron and nonlinear neuron is a nonlinear activation function. An example the activation function can be (often used), the binary step function (3).

$$y(x) = \begin{cases} 0 \; if \;\; x < a \\ 1 \; if \;\; x \geq a \end{cases} \tag{3}$$

where: a - activation threshold

A single neuron is a simple structure, easy to implement (both as software and as hardware). However, the neuron can't work alone. Neurons are found in greater numbers, as the neural network. The network consists of many interconnected neurons. The number of neurons depends on many factors:

- the number of input signals (in MLP networks, this is the number of neurons in the input layer)
- number of output signals (the number of neurons in the output layer);
- complexity of the problem - (number of neurons in the hidden layers).

A large network can be constructed of many hundreds or even thousands neurons. Method in which they are connected decide on the type and capacity (for applications) neural network. There are three main types of them:

- one way networks (feedforward),
- recurrent network,
- self-organizing maps.

The simplest one way network, called the single-layer network or single layer perceptron, composed of one layer of neurons (Figure 2a). Actually, two layers, but the first layer is not involved in the calculations - it is only a signal input. Number of neurons depends on the number of possible outcomes. Each neuron recognizes its own pattern of

inputs (it has its own set of weights). Neuron who recognized the pattern gives a positive answer on its output. All other neurons give a negative answer.

The simplest one way network, called the single-layer network or single layer perceptron, composed of one layer of neurons (Figure 2a). Actually, two layers, but the first layer is not involved in the calculations - it is only a signal input. Number of neurons depends on the number of possible outcomes. Each neuron recognizes its own pattern of inputs (it has its own set of weights). Neuron which recognized the pattern gives a positive answer on its output. All other neurons give a negative answer.

To solve more complex tasks, at least two-layer network structure are required. An example of this type of network is the multilayer perceptron (figure 2b).

Perceptrons constructed from layers of neurons with regular connections. Input of each neuron in a given layer is connected to the outputs of all neurons in the previous layer. The strength of connections between neurons depends on the weight. The number of layers and the number of neurons in the layer can be unlimited.

The first layer is called input layer, the last - the output layer. Between them there are hidden layers. Neurons in hidden layers act as intermediaries. They have access to the data input, and neurons in subsequent layers make decisions based on their results. The number of hidden layers can be unlimited, but because of the computation time is used mostly one or two hidden layers.

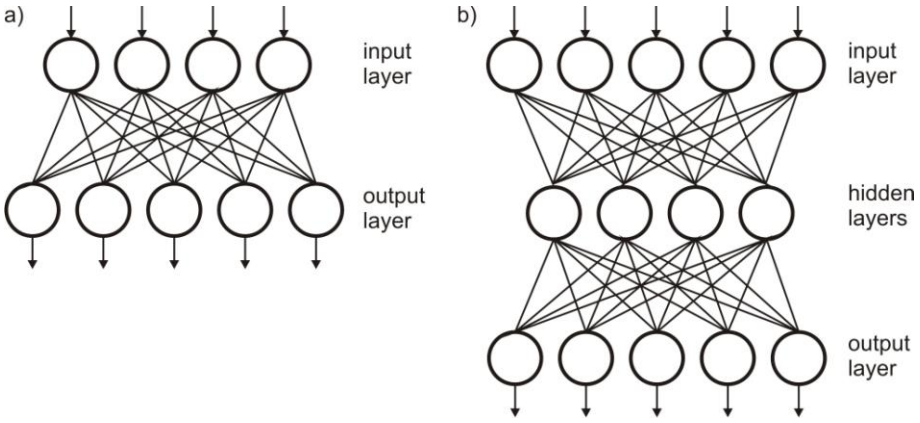

Fig. 2. a) one-layer network, b) two-layer network

One of classical types of artificial neural network is the recurrent Hopfield network. This network has a very simple learning algorithm and can be easily implemented in the form of a physical system.

Hopfield network consists of one layer containing N identical neurons. In this network there are feedback from the output of each neuron to the inputs of the other (Figure 3).

In this way, the i-th input neuron represents the i-th neural network, which is connected to the output of other neurons. The output of the ith neuron is also the i-th output of the network. The neural network are given distorted patterns. After completion of the network, verify that the obtained result is a sought model. The network can give a result valid, invalid and can not get the result, that is subject loop [4. p. 94-95].

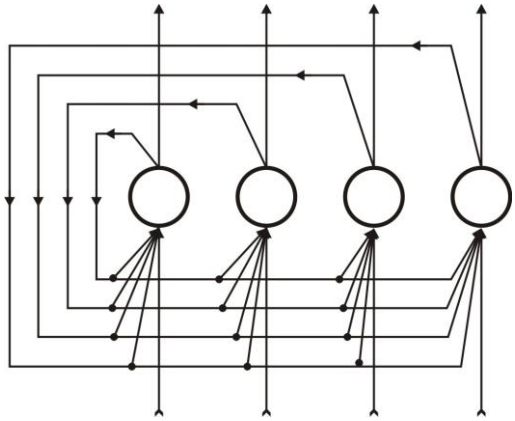

Fig. 3. Recurrent Hopfield network.

Operation cycle of neural network can be divided into two stages: learning and normal operation (solving tasks). The network starts its work with random weights of neurons. During learning these weights are modified, and the result error decreases. At the beginning of the learning process is very fast, but after a certain number of steps this speed decreases. Learning networks may need many thousands of steps. For each step is necessary to repeat the same calculation for each of the neurons.

One-learning network built from a few thousand neurons can take several to several tens of hours (on PC). There are two ways which can allow the widespread use of neural networks for PC computers. The first is the optimization of neural network algorithms (in particular learning algorithms), the second way is optimal use of PCs.

3. The possibility of a hardware implementation of artificial neural networks

Artificial neural networks can be implemented in hardware. Figure 4 shows the construction of analog-digital artificial neuron.

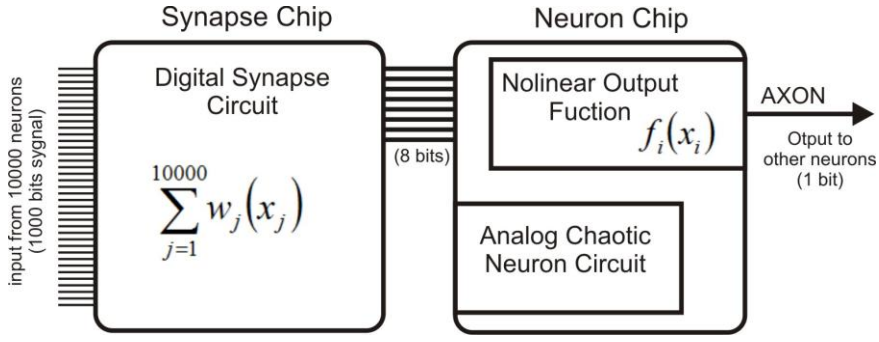

Fig.4. Diagram of analog-digital artificial neuron [5, p. 81]

It includes:

- Synapse Chip - input signals integrator with a 10 000 inputs, its task is to sum input neurons multiplied by the weight of the inputs (4).

$$e = \sum_{j=1}^{10000} w_j \left(x_j \right) \tag{4}$$

where: *w - the weight of input*
x - input signal

- Neuron chip - a block of non-linear activation function, equipped with 8-bit input (from Synapse Chip) and one bit output (Axon) to connect other neurons [5, p 80].

This kind of hardware solutions are being developed, but still for scientific purposes only. Very promising range of applications for systems of this kind is also education. The possibility of practical manipulation of artificial neurons, connecting them with different structures, can enable students to better understand how they work.

4. The specificity of the implementation of neural networks using a modern CPU architecture

Neural networks consist of many repeating elements - neurons. For each neuron perform the same calculations, but the input of these calculations (weights of neurons) are different.

In theory, all the neurons in the neural network are working together. In practice, the network is implemented (simulated) using a computer program, so the instructions are processed sequentially, one by one.

The first attempt was a parallel data processing pipelining. Processor instruction cycle can be divided, very simply, in several phases. For example:

- IF - instruction fetch - processor retrieves the command from the I-Cache (instruction cache - the cache block containing the instructions);
- ID – instruction decode - decoding unit recognizes the command, on the basis of generating control instructions for implementing the block, then initiates the download of arguments;
- EX – execute - during the implementation of an arithmetic logic unit performs the appropriate action;
- MA – memory access - processor retrieves data from the memory of D-Cache (data cache - the cache block containing the data);
- WB – write back - last step is to save the results [6, p. 24-25].

Each phase is performed by a separate unit within the processor. Five levels will split into five independent processor functional blocks - blocks that can operate simultaneously. Many instructions are processed at the same time, but still they are executed sequentially (in the order they are placed in the program). The processor works as a conveyor belt in a factory. Pipelining is not yet performing multiple instructions in parallel.

To be precise - the number of pipeline stages doesn't need be 5 as in the example above (this was the number of degrees in the Pentium processor.) In the next generation of processors, this number has changed (Table 1).

Table 1. Number of pipeline stages for the selected processor

Processor	Number of pipeline stages
Pentium	5
Pentium Pro	6
Pentium II	12
Pentium III	12
Pentium IV	24
Itanium (64 bits)	10

The next step in the development process was a superscalar processor. It allowed the completion of several orders in one tact clock. Superscalar processing was performed by placing a number in the processor execution units (arithmetic logic unit, floating point unit, etc.).

The first processor (on PC) executing processing of superscalar processor was a Pentium processor. It had two pipes (u-v-pipe and v-pipe). U-pipe was the main stream. He could perform all the instructions. V-pipe was a additional pipe . When two successive instructions could not have been paired (made parallel) worked only u-pipe pipeline.

Superscalar is a development of the processing pipeline. In modern processors, both of these techniques are used together. For example, the Pentium had two five-phase streams of data. A single-core superscalar processor is still classified as an SISD processor (Single Instructions, Single Data). The next step in the development of processors are MIMD processors (Multiple Instruction, Multiple Data streams) - multiple autonomous processors simultaneously executing different instructions on different data. This architecture is derived from the large computer clusters. In a multiprocessor computer systems, many tasks can be performed simultaneously (in parallel). Microcomputer now also use different architectures multiprocessor systems. Examples include HT Technology (Hyper Treading) and Multicore used in Intel processors [2, s. 332]. All modern multi-core superscalar processors family IA-64 are MIMD processors.

Modern multi-core processors allow you to create more complex neural networks for computers. This is possible by their high computational power.

Another possibility is the calculation of SIMD (Single Instruction, Multiple Data streams). A computer which exploits multiple data streams against a single instruction stream to perform operations which may be naturally parallelized. For example, an array processor or GPU. SIMD type processors are very efficient in some types of calculations. The use of these capabilities does not require special vector processors, but only the optimal use of existing technology. It is possible for ordinary PCs.

The artificial neuron has several inputs which are correlated with each other. The inputs of the neuron can be considered as a vector. The simultaneous calculation of the vector (or several vectors) can be done using multimedia extensions implemented on x86 processors.

Multimedia extensions have been introduced to better (faster) processing of multimedia. Data of this type (audio, video, film) have one thing in common. They consist of millions of repeated samples (sample of sound, image pixels, block of animated picture). Their process is a continuous perform of the same operation (one algorithm) for further information. Information provided as vectors.

The first of these extensions was the technology MMX (MultiMedia eXtensions) used in the Pentium [2, p. 11] processor. MMX is a set 57 SIMD instruction for Pentium and compatible. MMX commands can perform logic and arithmetic operations on integers. Each of the eight 64-bit MMX registers can contain multiple data (2 words of 32-bit, 4 words of 16-bit or 8 words and 8-bit). Program command is executed simultaneously on all the words contained in the registry. MMX instructions operate on vectors (understood as a one-dimensional arrays of numbers).

The next step in the development of multimedia extensions which was implemented in processor AMD technology 3DNow!. Unlike its predecessor command 3DNow! also operated on vectors of floating point numbers - architecture SIMD-FP (Single Instruction, Multiple Data streams - Floating Point). Intel introduced the architecture of the next generation of its multimedia extensions - SSE (Streaming SIMD Extensions). The development of SIMD instructions implemented in x86 processors is shown in the table.

Table 2. The development of SIMD instructions implemented in x86 processors.

Name	Changes in the architecture
MMX	SIMD instruction 57 (multiple data processing); 8 new 64-bit registers (shared with the floating point unit); Increased cache, instruction cache and instruction processing pipeline length;
3DNow!	SIMD-FP technology - vector calculations on floating-point numbers; 21 new instructions operate on 64-bit registers (later supplemented by further instructions to ensure compliance with SSE);
SSE	SIMD-FP technology, 48 new instructions, and eight 128-bit dedicated registers.
SSE2	144 new orders; integer calculations using 128-bit registers;
SSSE3	16 new instructions operate on vectors of integers
SSE4	54 new instructions operate on integers and floating-point;

The first multimedia extensions were designed as an alternative for graphics accelerators. Although the MMX and his successors have not replaced the GPU which are still developed. They are used, among other things, in processing and playback of multimedia. The ability to perform calculations on vectors (or several scalar simultaneously) accelerates the SIMD-type tasks. Learning neural networks belong to this type of tasks.

5. The specificity of the implementation of neural networks using a GPU

Even more opportunities in creating the neural networks are created by the using of GPU (Graphics Processing Unit). Powerful GPUs are present in all modern PCs. GPU is a

specialized electronic circuit designed to rapidly manipulate and alter memory in such a way so as to accelerate the creating of images in a frame buffer intended for output to a display. Modern GPUs are very efficient at manipulating computer graphics, and their highly parallel structure makes them more effective than general-purpose CPUs for algorithms where processing of large blocks of data is done in parallel.

Using unified shader technology enables you to use GPU not only for visual computing, but also for general computing (scientific, engineering, etc.). Examples of programming environments for such general calculations are:

- CUDA - Computer Unified Device Architecture developed by Nvidia; versatile multi-core processor architecture (mainly the GPU) enables the use of their computing power to solve general, numerical problems more efficient than the traditional, sequential processors, CUDA provides a programming language based on high-level programming environment C;
- BrookGPU - Brook for GPUs is a compiler and runtime implementation of the Brook stream program language for modern graphics hardware; The goals for this project are: demonstrate general purpose programming on GPUs; provide a tool for developers who want to run applications on GPUs; research the stream language programming model, streaming applications, and system implementations [7];
- OpenCL - Open Computing Language is a framework for writing programs that execute across heterogeneous platforms consisting of central processing unit (CPUs), graphics processing unit (GPUs), and other processors.

Although shaders were introduced for graphics related tasks, shaders can also be used for more generic computation, just as generic programs can be used to compute arbitrary data. As the computational power of GPUs continue to rise faster than conventional CPUs, the interest in shader programming attracts more and more attention. This requires rethinking algorithms or problems to fit the stream processing paradigm.

Example of using GPU shader to calculate the linear neuron in the OpenGL Shading Language (version 1.2 for OPGL 2.1) shown in Figure 4.

```
#version 120
  uniform vec3 weights
  uniform vec3 inputs

  void main(void)
  {
    gl_FragColor=vec4(dot(weights,inputs),0.0,0.0,1.0);
  }
```

Fig. 4. The procedure for realizing linear neuron using shader.

The code snippet in Figure 4 shows the method of implementation, using the shader, the scalar product of the neuron with three inputs. Weights and neuron inputs are given as vectors, the result as a component of the color point, that is calculated by the shader. Calculations of the three inputs are executed in parallel.

The procedure shown in Figure 5 shows method calculated, using the shader, the Euclidean distance between neuron input vector and weight vector.

```
#version 120
uniform vec3 weights
uniform vec3 inputs

void main(void){
  float d=(weights.x-inputs.x)*(weights.x-inputs.x);
  d=d+(weights.y-inputs.y)*(weights.y-inputs.y);
  d=d+(weights.z-inputs.z)*(weights.z-inputs.z);
  gl_FragColor=vec4(sqrt(d),0.0,0.0,1.0);
}
```

Fig. 5. The procedure for counting the Euclidean distance of two vectors using the shader.

In one-way networks shaders of your graphics card can perform calculations in parallel for all neurons layer. The self-organizing networks (Kohonen network, for example), all neurons can be computed in parallel.

Because modern graphics cards have hundreds, and often more than a thousand independent unified shaders their computing capabilities are very large.

6. Conclusions

Artificial neural networks is a technology developed for many years. They are successfully used in many fields of science, technology and economy, but their application often requires the use of supercomputers with very high computing power.

Provide opportunities offered by neural networks, the average computer user requires network implementation on personal computers. The problem is in a great demand these kinds of programs on the computing power.

You can specify three possible ways to solve this problem:

- optimization of neural network algorithms,
- designing and building low-cost (generally available) hardware (neuroprocessors)
- optimal use of technology developed for personal computers.

The last working methods described here seems to be most promising. The development of personal computers, especially SIMD technologies used in modern GPU can allow for the implementation of a neural network with a high degree of complexity by using a commercially available and inexpensive equipment.

References

[1] TADAEUSIEWICZ R.: *Sieci neuronowe*. Akademicka Oficyna Wydawnicza RM, Warszawa 1993.

[2] STANISŁAWSKI W., RACZYŃSKI D.: *Programowanie systemowe mikroprocesorów rodziny x86*, Wydawnictwo Naukowe PWN, Warszawa 2010.

[3] TADEUSIEWICZ R.: *Elementarne wprowadzenie do techniki sieci neuronowych z przykładami*, Akademicka Oficyna Wydawnicza PLJ, Warszawa 1998.
[4] KORBACZ J. OBUCHOWICZ A. ,UCIŃSKI D.: *Sztuczne sieci neuronowe*, Akademicka Oficyna wydawnicza PLJ, Warszawa 1884.
[5] Alrytnow M., A., Gołyszkin A., N., Kazncew P., A., Ostanienko N.: *Neurokomputery*, Moskawa 2008 (in rusian: АЛЯУТПОИНВ М. А.: ГАЛУШКИН, КАЗАНЦЕВ П. А., ОСТАПЕНКО Г. П., *НеирокомпьюТеры*, Горячая линия – Телеком, Москва 2008).
[6] METZGER P.: *Anatomia PC, Wydanie XI*, HELION, Warszawa 2007.
[7] http://graphics.stanford.edu/projects/brookgpu/

SPRZĘTOWE MOŻLIWOŚCI WSPIERANIA TWORZENIA SZTUCZNYCH SIECI NEURONOWYCH

Streszczenie

Artykuł poświęcony jest możliwości sprzętowego wspomagania tworzenia sztucznych sieci neuronowych. Autor przedstawia w nim możliwości budowy i sprzętowych neuro-procesorów, możliwości wykorzystania instrukcji typu SIMD zaimplementowanych w nowoczesnych procesorach oraz zunifikowanych shaderów procesora graficznego. Autor przedstawia przykład prostej implementacji pojedynczego neuronu przy użyciu shadera karty graficznej.

Słowa kluczowe: sztuczne sieci neuronowe, SIMD, GPU, shader