A Survey of Neuro-inspired Computing Systems

The H. Vu
The University of Aizu,
Graduate School of Computer Science and Engineering,
Adaptive Systems Laboratory,
Fukushima-ken, Aizuwakamatsu-shi 965-8580, Japan
E-mail: d8182106@u-aizu.ac.jp

Abstract

Artificial Neural Networks (ANNs) are exposing great abilities to address a huge variety of problems in areas such as image processing, pattern recognition, and medical diagnostics. ANNs are parallel and distributed systems for information processing. This characteristic is suitable for implementing ANN systems in hardware architectures because of the advantages of their inherent parallelism and their potential to meet the demands of real-time applications. This brief survey compares the most common solutions which are used to implement ANN systems. Furthermore, an overview of ANN systems is also presented.

1 Neural Network Fundamental

1.1 Neural models

A coarse biological neuron, shown in Figure 1 (a) is considered to be an information processing system. *Dendrites* play a role as input devices, where input signals are collected. The neuron will process the signals and then produce output signals which are propagated along its *axon*. Finally, the axon transmits the signals via synapses to dendrites of other neurons. It is important to emphasize that this model of a biological neuron is very coarse, and there are many different types of neurons, each of which has different properties.

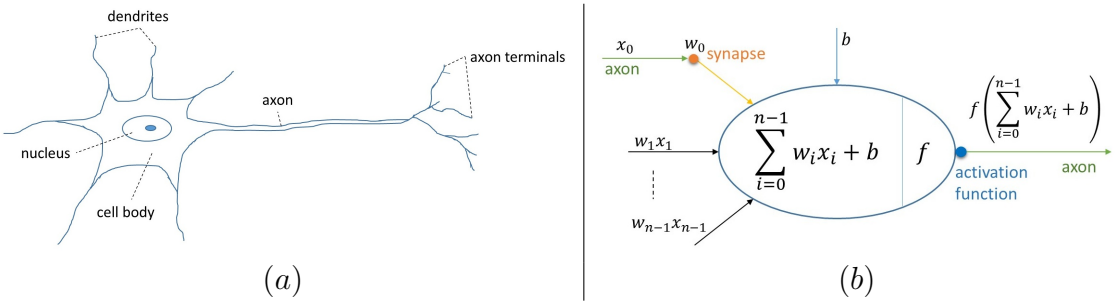

Figure 1: (a) a cartoon drawing of a biological neuron (b) a mathematical model of a neuron.

In the computational model of a neuron, shown in Figure 1 (b), each output signal (e.g. x_0) from a previous neuron is multiplied with a weight (e.g. w_0). This weight presents the synaptic strength at that synapse. Dendrites carry the signals (e.g. w_0x_0) to the cell body where they

all get summed. In the basic model, if the final sum exceeds a certain threshold, the neuron can *fire*, sending a spike along its axon. An activation function is modeled as the *firing rate* of the neuron.

Neural networks can be classified into three generations according to their computation units [18, 19]:

The first generation: networks based on neurons as computational units which are also referred to as perceptrons or threshold gates. These networks only process with digital inputs and outputs, boolean functions, and a single hidden layer. Multilayer perceptrons, Hopfield networks, and Boltzmann machines are typical examples of this kind of neural network.

The second generation: each neuron in the network applies an activation function with a continuous set of possible output values, such as sigmoid or polynomial or exponential functions. Feedforward, recurrent sigmoidal neural networks, and radial basis function units are considered typical examples of this generation. Moreover, these systems compute not only arbitrary boolean functions but also functions with analog inputs and outputs. Furthermore, neural networks in this generation support learning algorithms based on gradient descent.

The third generation: spiking neurons are employed as computational units in the network. Spiking neural networks are considered to be a closer approach to modeling biological neurons than previous ANNs. Biological neural systems use the timing of single action potentials (or spikes) to encode information. SNNs use the timing of the spikes, the network topology, and synaptic weights to process information.

1.2 Neural Network Architectures

There are many neural network architectures; they are regularly organized by their different layers of neurons. These layers comprise of input, hidden, and output layers. Two metrics are frequently used to measure the neural network size, the number of neurons and the number of parameters. For example, there are ten neurons in the network showed in Figure 2 (not counting neurons in the input layer), while the number of parameters is 46 (3x4 + 4x4 + 4x2 = 36 weights, 4+4+2 = 10 bias)

It is important to mention that the network size plays an integral part in designing a neural network. Designers need to decide the number of layers in the network, and the number of neurons in each layer. When the network size is increased, it means that capacity of the network also is increased. This results in the network is ability to handle more complicated functions. On the other hand, overfitting can occur when increasing the network size. However, there are some good techniques to deal with this problem, such as L2 regularization, dropout, and input noise.

1.2.1 Feed forward neural networks

Figure 2 depicts a **feed forward neural network (FF or FFNN)**. This network is organized from separate layers of neurons, they are input, hidden, and output layers. In this architecture, there are many connections between neurons across layers, but not within a layer. Information is fed from the front to the back. This network usually is used with the back-propagation training method.

There are some other neural networks with the same topology as FFNNs. If neurons use a simple binary function, this architecture is called **Perceptron (B) or Multilayer perceptron (MLP)**. The simplest network, with two input neurons and one output neuron, can be used to model logic gates. **Radial basis function (RBF)** [4] networks are FFNNs with neurons having radial basis functions. RBFs are suitable for pattern recognition and classification.

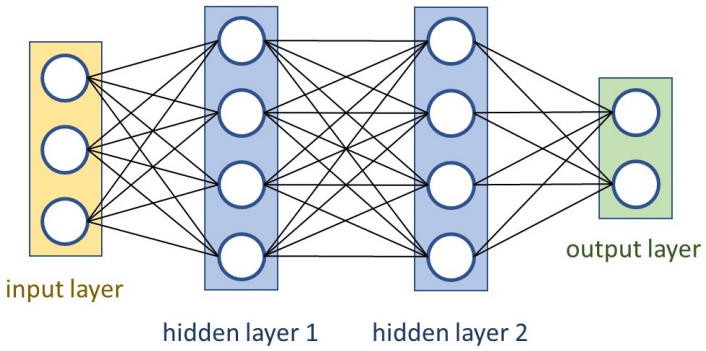

Figure 2: A feed forward neural network.

1.2.2 Recurrent neural network architectures

As shown in Figure 3, **Recurrent neural networks (RNNs)** are pretty similar to FFNNs, but the hidden layers are replaced by recurrent neural layers. Unlike FFNNs, Neurons in RNNs are not only fed from the previous layer but also from the previous pass of themselves [8]. This results in different outputs when changing the order of information in feeding.

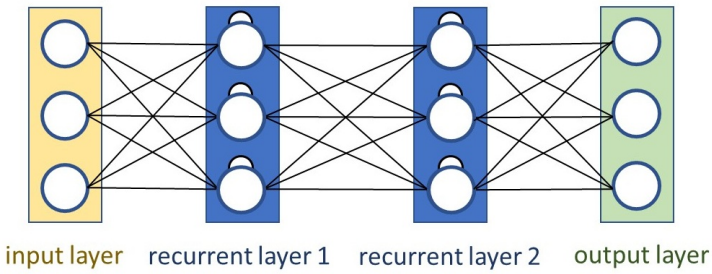

Figure 3: A recurrent neural network.

RNNs can be used in many areas where the data form can be represented as a sequence such as a string of text. Thus, RNNs are regularly used in autocompletion systems and machine translation. A big drawback of RNNs is the vanishing/exploding problem when using gradient descent technique.

Long short term memory (LSTM) [13] networks are proposed to deal with the problem of RNNs which were mentioned above. Instead of using recurrent neurons, LSTM networks employ memory neurons which are composed of a memory cell and three gates: input, output and forget. One advantage of LSTMs is that the network can learn complex sequences or compose primitive music. However, LSTMs require more resources to implement compared to RNNs because each gate in LSTM neuron has a weight.

In order to tackle the drawback of LSTMs, **Gated recurrent units (GRUs)** [7] were proposed. Each GRU neuron has only two gates: update and reset gates. GRUs are slightly faster and easier to run compared to LSTMs.

Neural Turing machine (NTM) [10], as shown in Figure 4, is another proposal based on LSTMs. Each NTM neuron does not comprise a memory cell. The network uses a content-addressable memory bank which can be read and written. NTMs have been shown to be able to read, write, and even change state based on what it read.

Other proposals are **Bidirectional recurrent neural networks (BiRNN), bidirectional long short term memory networks (BiLSTM)** and **bidirectional gated recurrent units (BiGRU)**. Unlike their counterparts, instead connecting to the past, they

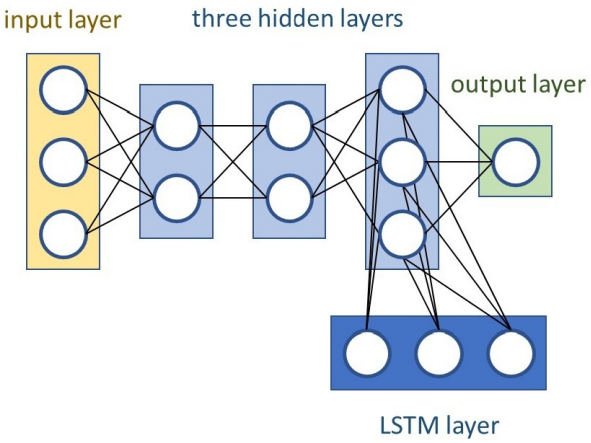

Figure 4: A neural Turing machine network.

connect to the future. In [24], BiRNNs provided an efficient estimation of conditional posterior probability of complete symbol sequences without making any explicit assumption about the distribution shape.

1.2.3 Convolutional neural network architectures

Convolutional neural networks (CNNs or deep convolutional neural networks, DCNNs) are very suitable for image processing [17]. They can also be used for other input types such as audio. CNNs sequentially feed each part of training data to decrease the number of nodes in the network. CNNs regularly use the back-propagation technique in training.

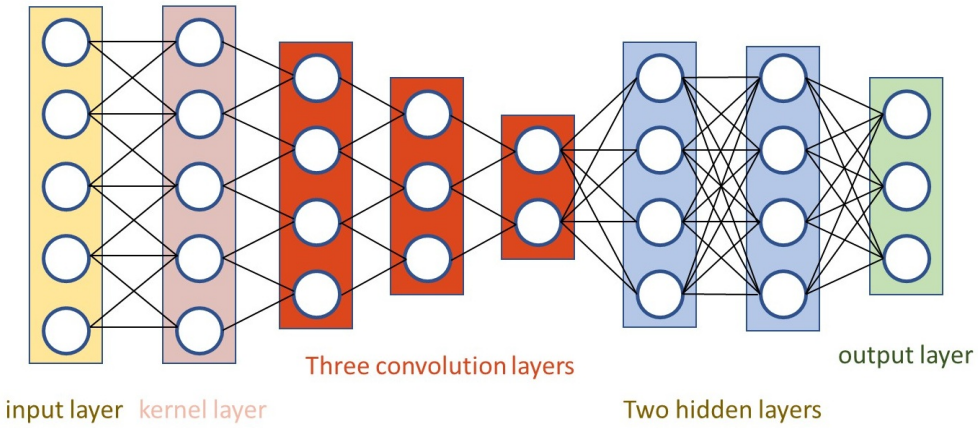

Figure 5: A convolution neural network.

A CNN network is regularly composed of a few different types of layers: convolutional layer, pooling layer, and fully-connected layer. *The convolutional layer* is the core block of a CNN that does most of the computations. This layer will compute a dot product between their weights (kernels or filters) and a small region of input volume. *The pooling layer*, if it is added to the CNN, will perform a downsampling operation in order to decrease the spatial dimensions. 2x2 max poolings are commonly used in image processing. *The fully-connected layer* will compute and produce the outputs.

The are some other proposals based on CNNs. **Deconvolutional Networks (DNs)** [25], as shown in Figure 6, were proposed to get a robust image representation for both the analysis

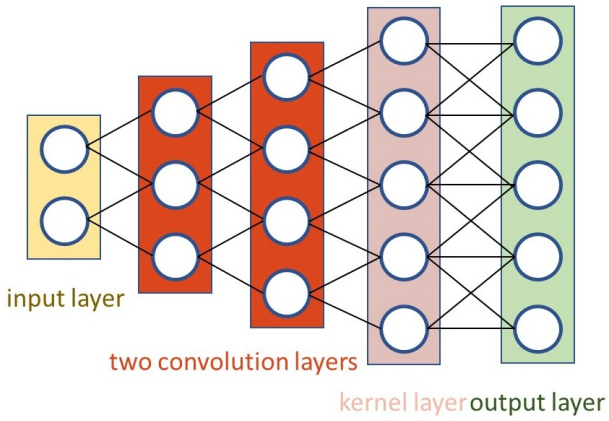

Figure 6: Example of a deconvolutional neural network.

and synthesis of images. **Deep convolutional inverse graphics network (DC-IGN)** [16] is built convolutional layers and de-convolution operators. This model was shown to be able to generate new images of the same object with variation in pose an lighting when given a single input image.

1.2.4 Autoencoder architectures

Autoencoders (AE) [3] were proposed to encode information in term of compression automatically or reduce dimensions of the feature space in information processing applications. Figure 7 (a) depicts a autoencoder, it is quite similar to a FFNN. However, there are two characteristic which need to be mentioned. First, the number of neurons in middle layers are less than other layers in the network. This is because the information is compressed in the middle layers. Second, the AE architecture is symmetrical around the middle layers.

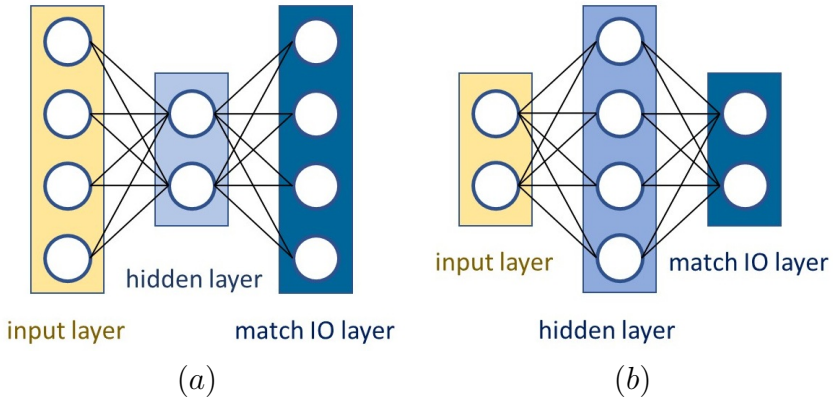

Figure 7: Examples of (a) autoencoder (b) sparse autoencoder.

Sparse autoencoders (SAEs) [23] are contrary to AEs, as shown in Figure 7 (b). Compared to AE architectures, SAE architectures are still symmetric, but the number of neurons in the middle layers is larger than its counterparts. SAEs can be used to extract small features from a dataset.

1.2.5 Hopfield network and Boltzmann machines

Hopfield networks(HNs) [14] are quite different compared to the systems mentioned

above. In this architecture, each neuron is connected to others. Neurons play distinct roles when the network is trained, they are input, hidden, and output corresponding to before, during, and after training respectively. HNs offer a model for understanding human memory. They are also used as content-addressable memories.

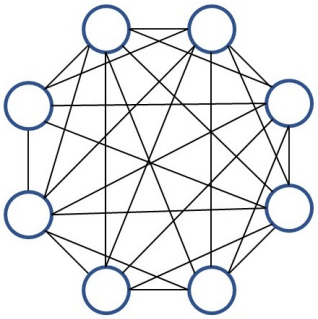

Figure 8: Example of a Hopfield neural network.

Boltzmann machines (BMs) [12] are pretty similar to HNs. However, BMs are composed of some input neurons, while the others are hidden neurons. The input neurons will become output neurons after each update of the full network.

2 Neural-Inspired Computing Systems

An ANN system is a parallel and distributed network of neurons which are interconnected in a layered arrangement. This section provides brief summaries of implementing ANNs on several hardware platforms.

2.1 ASIC Analog SNN/ANN systems

There are several good reasons why designers should implement ANNs into ASIC analog chips. First, developers can build some common activation function units quickly. For example, functions such as integration and sigmoid are implemented by using single transistors. Second, ASIC analog chip can be automatically performed by physical processes such as summing of currents or charges. Finally, analog chips offer area, speed, and power consumption benefits.

On the other hand, ASIC analog SNN/ANN systems face some challenges. The first one is that analog technology is susceptible to noise. This limits computation precision and makes it harder to understand what is computed exactly. The second one is how to represent adaptable weights in analog circuits. Weights can be represented by resistors, but these are not adaptable after the production of the chips. Besides, capacitors, floating gate transistors and charge coupled devices can be used for weight representation, but they have limited storage times and lack compatibility with standard VLSI processing technology. The final one is that implementation of most learning rules into analog VLSI turns out to be very difficult.

2.2 ASIC Digital SNN/ANN systems

The majority of the ASIC chips are digital, and most of them use CMOS technology. Digital techniques offer high computation precision, high reliability, and programmability. Furthermore, there are powerful design tools which are available for fully and semi-digital system designs.

However, ASIC digital chips have some drawbacks. First, ASIC digital chips are lower speed compared to their analog counterparts because it is difficult for designers to implement the synapse multiplier. This element is normally the slowest in the network processing. Second, the chips have may not directly interface with the real, analog word. Digital implementation always requires fast analog-to-digital converters (ADCs) in the input port and digital-to-analog converters (DACs) in the output port. This leads to an increase in the cost of implementation and power consumption, as well as decreases throughput of systems.

2.3 FPGA SNN/ANN systems

FPGAs offer an effective programmable resource for implementing SNN/ANN systems. They allow different designs to be evaluated in a very short time. Implementing neuron networks on FPGA has some other advantages such as: low cost, readily available, and flexibility. Therefore, it is easier for designers to change their designs and also reduce the hardware development cycle. On the other hand, one main drawback of this implementation method is that it is harder to implement large models with thousands of neurons; however, FPGA speeds approximately double each year. Thus, a large neural network may soon be implemented on a single FPGA.

2.4 DSP SNN/ANN systems

Like microprocessors, Digital Signal Processors (DSP) are not suitable for ANN systems because the main disadvantage with DSP implementation is sequential computation. However, DSPs can be used as computational units when implementing ANN systems. In this case, a DSP is designed as a specialized microprocessor that is optimized for the fast operational needs of digital processing. For example, in [22], DSP cores built in a FPGA chip are used for product implementations. As mentioned in Section 2.2, dot product implementations are difficult tasks for designers. In this architecture, the DSP is designed to use a frequency twice as large as a neuron block. As a result, a product operation can be done in one operation cycle of the FPGA. In another work, a floating point DSP processor is used to implement a dynamic synapse neural network in order to recognize acoustic sound in noisy environments.

Digital chip can be classified into three main categories: like bit-slide, single instruction multiple data (SIMD), and systolic arrays. In the case of conventional bit-slice architectures, a processor is built from modules. Each module processes one bit-field or "slide" of an operand. This architecture offers simple and cheap building blocks of single or several neurons to construct larger size networks. This make chip designing become more flexible. Some examples of this architecture are Philips' Lneuro chip [20] and Neuralogix' NLX-420 Neural Processor. Chips based on this architecture often offer off chip learning. In the SIMD, Processing Elements (PEs) concurrently run the same instruction with different data sets [21]. Adaptive Solutions' N64000 [11] and SIMD architecture [15] are good examples for this architecture. In the case of systolic array architectures, each PE processes one stage of a calculation. This architecture is very suitable for implementing matrix multiplication that is common to ANNs. Some examples are TORAN [2] and vector processor arrays [6].

2.5 Examples

The **TrueNorth** chip [1] is based on an efficient, scalable, and flexible non-von Neumann architecture. It is built from 4096 neurosynaptic cores, containing an aggregate of 1 million neurons and 256 million synapses. Chips can be tiled in two dimensions via an interchip communication interface. TrueNorth can reach a peak computational performance of 58 giga-synaptic operations per second (GSOPS) and an energy efficiency of 400 GSOPSW [5]. The

architecture of the chip is suitable for many applications that use complex neural networks in real time such as multiobject detection and classification.

Another remarkable architecture is **SpiNNaker** [9], a massively parallel multicore computing system. It can simulate the behavior of aggregates of up to a billion neurons in real time. The full architecture is composed of 1,036,800 ARM9 cores and 7 Tbytes of RAM in 57K nodes. The system consists of 1,200 boards where each of them contains 48 nodes. In operation, the system consumes at most 90 kW of electrical power.

2.6 Conclusion

As mentioned above, there are several solutions which are commonly used to implement ANN systems in hardware. Each of them shows both advantages and disadvantages, and choosing one depends on some factors such as speed, area, cost, design time, and reliability. Regarding speed and area, ASIC analog systems are considered to be the best choice for the ANN implementation; however, these systems have some disadvantages, with high cost, large design time, and low reliability. ASIC digital systems are more reliable compared to analog counterparts, but they are slightly slower and smaller compared to analog ASIC systems in terms of speed and area. FPGA implementations show a balance between the factors. Even though they have slower speed and higher area than ASIC technologies, they are very favorable regarding cost, design time and reliability. In addition, DSP chips are a good choice if design time, cost and reliability are the top priority of implementation.

The survey also presented a brief overview of several ANN architectures, each of them had both advantages and drawbacks. It is difficult to choose one which is suitable for many applications. Additionally, there are many other ANNs which are not mentioned this paper.

References

[1] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G. J. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(10):1537–1557, Oct 2015.

[2] H. Amin, K. M. Curtis, and B. R. Hayes Gill. Two-ring systolic array network for artificial neural networks. *IEE Proceedings - Circuits, Devices and Systems*, 146(5):225–230, Oct 1999.

[3] H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biol. Cybern.*, 59(4-5):291–294, September 1988.

[4] David S. Broomhead and David Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. *Royal Signals and Radar Establishment Malvern (United Kingdom)*, 1988.

[5] A. S. Cassidy, R. Alvarez-Icaza, F. Akopyan, J. Sawada, J. V. Arthur, P. A. Merolla, P. Datta, M. G. Tallada, B. Taba, A. Andreopoulos, A. Amir, S. K. Esser, J. Kusnitz, R. Appuswamy, C. Haymes, B. Brezzo, R. Moussalli, R. Bellofatto, C. Baks, M. Mastro, K. Schleupen, C. E. Cox, K. Inoue, S. Millman, N. Imam, E. Mcquinn, Y. Y. Nakamura, I. Vo, C. Guok, D. Nguyen, S. Lekuch, S. Asaad, D. Friedman, B. L. Jackson, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha. Real-time scalable cortical computing at 46 giga-synaptic ops/watt with 100 speedup in time-to-solution and 100,000 reduction in

energy-to-solution. In *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 27–38, Nov 2014.

[6] Jai-Hoon Chung, Hyunsoo Yoon, and Seung Ryoul Maeng. A systolic array exploiting the inherent parallelisms of artificial neural networks. *Microprocess. Microprogram.*, 33(3):145–159, May 1992.

[7] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[8] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[9] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown. Overview of the spinnaker system architecture. *IEEE Transactions on Computers*, 62(12):2454–2467, Dec 2013.

[10] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014.

[11] D. Hammerstrom. A vlsi architecture for high-performance, low-cost, on-chip learning. In *1990 IJCNN International Joint Conference on Neural Networks*, pages 537–544 vol.2, June 1990.

[12] G. E. Hinton and T. J. Sejnowski. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning and Relearning in Boltzmann Machines, pages 282–317. MIT Press, Cambridge, MA, USA, 1986.

[13] Sepp Hochreiter and Jrgen Schmidhuber. Long short-term memory. *Neural computation 9.8*, 1997.

[14] J. J. Hopfield. Neurocomputing: Foundations of research. chapter Neural Networks and Physical Systems with Emergent Collective Computational Abilities, pages 457–464. MIT Press, Cambridge, MA, USA, 1988.

[15] Dongsun Kim, Hyunsik Kim, Hongsik Kim, Gunhee Han, and Duckjin Chung. *A SIMD Neural Network Processor for Image Processing*, pages 665–672. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[16] Tejas D. Kulkarni, William F. Whitney, Pushmeet Kohli, and Joshua B. Tenenbaum. Deep convolutional inverse graphics network. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS'15, pages 2539–2547, Cambridge, MA, USA, 2015. MIT Press.

[17] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[18] Wofgang Maas. Networks of spiking neurons: The third generation of neural network models. *Trans. Soc. Comput. Simul. Int.*, 14(4):1659–1671, December 1997.

[19] Wolfgang Maass. On the relevance of time in neural computation and learning. In *Proceedings of the 8th International Conference on Algorithmic Learning Theory*, ALT '97, pages 364–384, London, UK, UK, 1997. Springer-Verlag.

[20] N. Mauduit, M. Duranton, J. Gobert, and J. A. Sirat. Lneuro 1.0: A piece of hardware lego for building neural network systems. *Trans. Neur. Netw.*, 3(3):414–422, May 1992.

[21] R. W. Means and L. Lisenbee. Extensible linear floating point simd neurocomputer array processor. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, volume i, pages 587–592 vol.1, Jul 1991.

[22] Francisco Ortega-Zamorano, Jose M. Jerez, Daniel Urda Munoz, Rafael M. Luque-Baena, and Leonardo Franco. Efficient Implementation of the Backpropagation Algorithm in FPGAs and Microcontrollers. *IEEE Transactions on Neural Networks and Learning Systems*, 27(9):1840–1850, 2016.

[23] Christopher Poultney, Sumit Chopra, and Yann Lecun. Efficient learning of sparse representations with an energy-based model. In *Advances in Neural Information Processing Systems (NIPS 2006*. MIT Press, 2006.

[24] Mike Schuster and Kuldip K. Paliwa. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[25] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2528–2535, June 2010.