PHILOSOPHICAL TRANSACTIONS A

rsta.royalsocietypublishing.org

Research

Article submitted to journal

Subject Areas:
artificial intelligence, electrical engineering, microsystems

Keywords:
neural network, handwritten digit, classification, time-mode, energy efficiency, ultra-low energy

Author for correspondence:
Omer Can Akgun
e-mail: o.c.akgun@tudelft.nl

An Energy Efficient Time-Mode Digit Classification Neural Network Implementation

O. C. Akgun[1] and J. Mei[2]

[1]Section Bioelectronics, Department of Microelectronics, Delft University of Technology, the Netherlands

[2]Department of Neurology and Department of Experimental Neurology, Charité - Universitätsmedizin Berlin, Germany

This paper presents the design of an ultra-low energy neural network that utilizes time-mode signal processing (TMSP). Handwritten digit classification using a single-layer artificial neural network (ANN) with a Softmin-based activation function is described as an implementation example. To realize time mode operation, the presented design makes use of monostable multivibrator based multiplying analog-to-time converters, fixed-width pulse generators and basic digital gates. The time-mode digit classification ANN was designed in a standard CMOS 0.18 μm IC process and operates from a supply voltage of 0.6V. The system operates on the MNIST database of handwritten digits with quantized neuron weights and has a classification accuracy of 88%, which is typical for single-layer ANNs, while dissipating 65.74 pJ per classification with a speed of 2.37k classifications per second.

1. Introduction

Machine learning is the study of models and algorithms that give rise to generalizable understanding of data and task completion without explicitly programmed instructions. As one of the many approaches in machine learning, artificial neural networks (ANNs) are partly inspired by connectivity and property of biological neurons and have proven to achieve considerable performance in a number of application areas.

THE ROYAL SOCIETY PUBLISHING

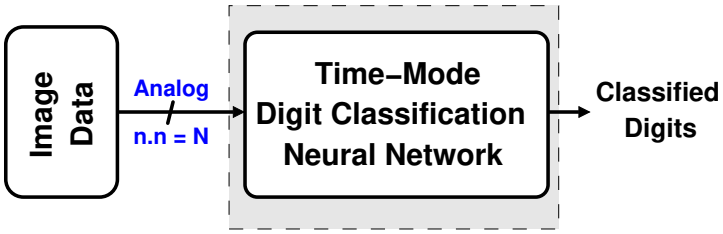


Figure 1. Proposed TMSP ANN high-level block diagram.

These areas include machine translation [1,2], computer vision [3,4], pattern recognition [5–7], game-playing [8,9] and medical diagnosis [10,11]. In the applications that require real-time operation, e.g., speech [5] and human action [7] recognition, and physical activity and patient monitoring [12], there is a need for always-on sensing. However, one of the challenges of the modern machine learning algorithms is their energy dissipation [13]. Most of the machine learning hardware development is done using either standard cell digital design methods [14,15] or mixed-signal methods [16] employing analog processing techniques in CMOS technologies.

The advancement and scaling of CMOS technologies have always been based on improving the performance of digital systems. With each new technology node, the threshold voltages of the available MOS transistors and the supply voltage of the process node is scaled as well. Scaling of the supply voltage reduces the headroom that is available to the transistors for operating in the saturation region. Without transistors operating in the saturation region, it is very hard to realize signal processing and amplification functions in the analog domain. One solution to this problem is using time-mode signal processing (TMSP) techniques [17–19]. Time-mode (TM) circuits represent an analog signal by the time difference between two binary switching events. Furthermore, when compared to standard digital design practices, time-mode operation is inherently of lower power. For example, when compared to standard CMOS digital circuit operation, to transfer N bits of data, the number of switchings required may change from 0 to N on the data line if the data is transmitted in parallel, whereas in a time-mode circuit transfer of the data always takes two switchings if the rising and falling edges of a pulse is used for information transmission. There are other advantages of TM operation, especially for machine learning hardware implementations: i) time-mode operation allows the designer to reduce the supply voltage and still realize analog-like functions, as will be shown in this paper, and ii) using single wires for data transmission instead of using data buses will allow a hardware designer to realize densely connected ANNs on chip more easily. Based on these observations, it is arguable that more low-power signal processing and machine-learning systems will be implemented using time-mode signal processing techniques in the future.

The research work presented here focuses on developing a time-mode digit-classification single-layer neural network for ultra-low energy operation. The proposed system is shown in Figure 1. A digit classification ANN was chosen for its simplicity, and well studied and understood behavior. During the design and training of the ANN, image data from a widely available dataset, MNIST [20], was used. n by n image data was converted into analog values and applied to the TM ANN. The applied image data is processed by the TM ANN by accumulating weighted delay values and a classification signal for the input image is generated. As it will be presented in the paper, TMSP allows such an ANN to work with extremely low energy dissipation values and with classification accuracy that is typical for single-layer ANNs.

Contributions of this paper are as follows. A time-mode implementation of a handwritten digit classification ANN is presented. Optimization steps for both system level and hardware level design are given, followed by the details of sub-block designs. The designed ANN is verified by both system level mathematical simulations as well as with transistor level SPICE simulations. The design is characterized for classification accuracy, energy dissipation and

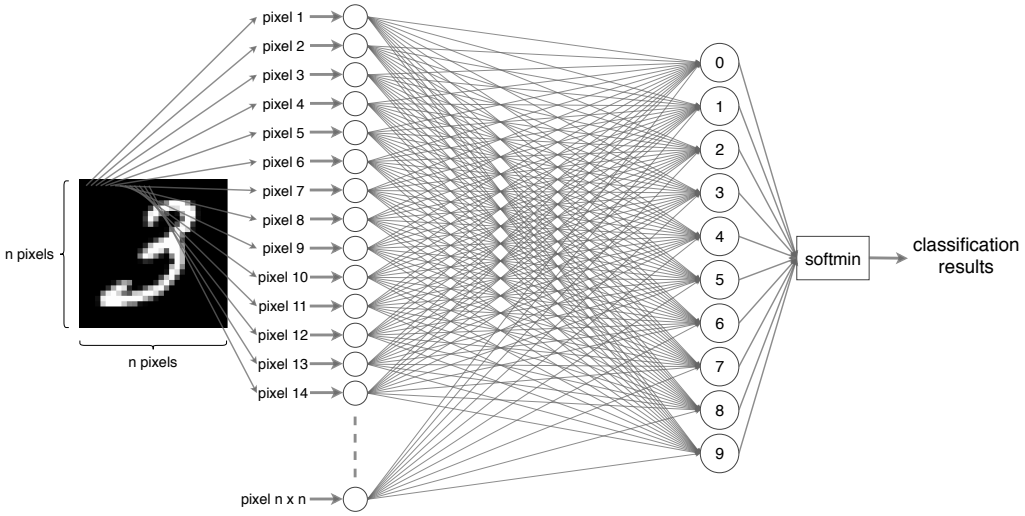


Figure 2. A fully-connected, single-layer neural network receiving inputs from an image of handwritten digit 3.

classification speed. The organization of the paper is as follows. Section 2 presents the high-level details and implementation steps of the ANN in software. Section 3 describes the TMSP ANN implementation with sub-block design and performance improvement steps. Transistor level simulation results are presented in Section 4 together with performance metrics, and finally the conclusions are drawn in Section 5.

2. Artificial Neural Network

In the present study, we implemented a hardware version of a time mode, fully-connected, single-layer neural network to recognize handwritten digits (Figure 2), using the MNIST database of handwritten digits [20]. The MNIST database contains a training set of 60,000 images and a test set of 10,000 images, with all images of size 28×28 pixels.

As presented in Figure 2, a single layer of neurons applying a linear transformation to the input data (i.e., MNIST handwritten digits) was constructed. The size of input sample was set to 784 (MNIST handwritten digits of 28×28 pixels) with an input range [0, 1] and the size of output sample was set to 10 (10 possible output digits ranging from 0 to 9).

An artificial neuron in the implemented ANN receives pixel data from input units (Figure 3). Each input $x_1 ... x_n$ is multiplied by its respective weight $w_1 ... w_n$, and the artificial neuron receives and sums all weighted inputs according to:

$$\begin{aligned} f(X) &= w_1 x_1 + w_2 x_2 + ... + w_i x_i + ... + w_n x_n \\ &= \sum_{i}^{n} w_i x_i \end{aligned} \tag{2.1}$$

Afterwards, an activation function is used to process the weighted sum. In the presented implementation, Softmin activation function, which takes all the weighted sums as input, is used (Figure 2). By processing the weighted sums, Softmin function rescales and assigns probabilities to the classified digit outputs. As a result of the Softmin function, each output is squashed to a value in the range $(0, 1)$ and the sum of all outputs add up to 1. The Softmin function is defined as:

$$Softmin(x_i) = \frac{exp(-x_i)}{\sum_j exp(-x_j)} \tag{2.2}$$

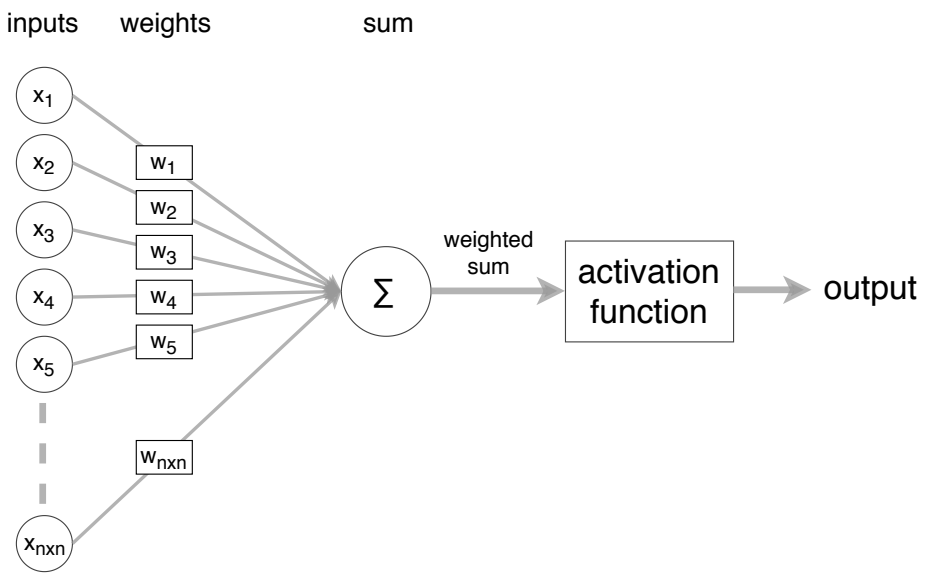


Figure 3. An artificial neuron with $n \times n$ inputs. Each input is associated with a weight. The weighted sum of all inputs is transformed by the activation function to produce the output.

After the weighted sums of the inputs are transformed by the activation function, final classification results are supplied by the ANN, i.e., if the n-th value is the highest in the output vector of the Softmin function, it means that weighted sum output of the n-th neuron is the smallest and n-th neuron has the highest probability to successfully classify the input to be digit n. Unlike most of the implementations that used Softmax as the activation function, in the present study, Softmin was used. This is based on the assumption that with Softmax, an artificial neuron with a greater weighted sum (i.e., greater accumulated delay in the hardware implementation) wins. However, this does not correspond to the targeted hardware implementation in which the fastest neuron is favored and therefore, using a Softmax activation function would decrease the operating speed of the ANN. Hence, we have chosen the Softmin activation function with which the fastest neuron wins and classification speed of the ANN is higher.

The described high-level ANN was implemented, trained and tested on the PyTorch framework [21]. The training of the ANN was done using batches of 100 images per batch and for 10 epochs. During the off-line training of the ANN, floating point values were used, however during hardware implementation and high-level verification simulations, the weight values were scaled to a range and quantized.

The adaptive moment estimation method (Adam) [22] was used as the optimization method during the training of the ANN. Adam optimization combines the advantages of both Adaptive Gradient Algorithm (AdaGrad [23]), which works well with sparse gradients, and Root Mean Square Propagation (RMSProp [24]), which works well in on-line and non-stationary settings. Kingma and Ba [22] suggested that instead of generating its parameter updates using a momentum (like RMSProp with momentum does), updates of Adam may be directly estimated using an average of first and second moment of the gradient. As a result, Adam performed equal or better than RMSProp regardless of hyperparameter setting. In L2-regularized multi-class logistic regression, Adam converged faster than AdaGrad. In a dataset with sparse features, Adam converged as fast as AdaGrad while dealing with space features efficiently. In an experiment with convolutional neural networks, Adam converged considerably faster than AdaGrad. For a more detailed discussion, see [22,25]. During training, the learning rate (α) was set to 0.01 while all other parameters were configured based on the default settings recommended in [22] ($\beta_1 = 0.9$,

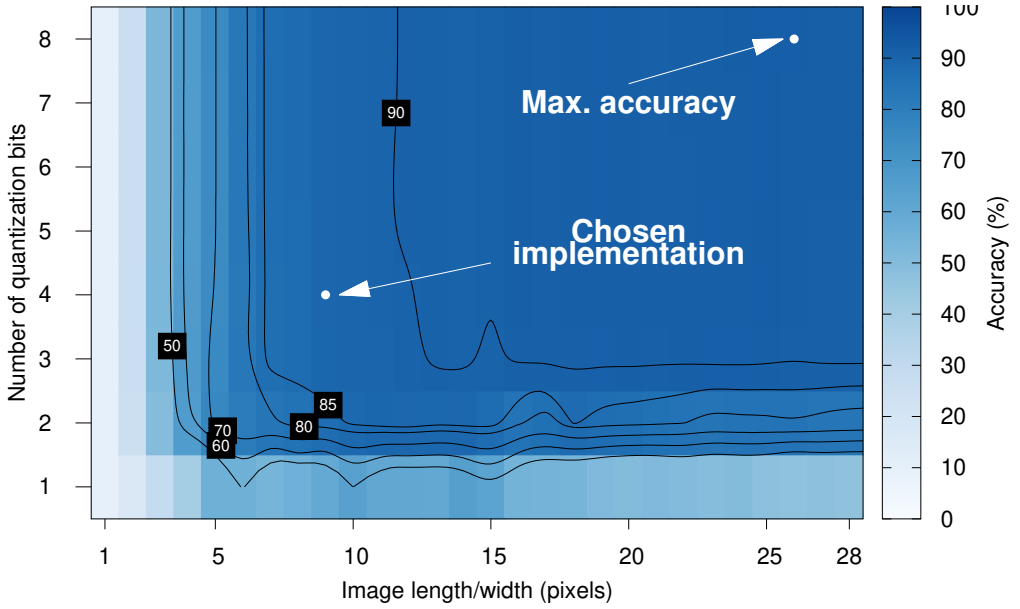


Figure 4. Classification accuracy of the designed ANN for varying image length/width sizes and weight quantization bits. Accuracy contour lines are also drawn for easier reference. Maximum accuracy and chosen implementations are marked.

$\beta_2 = 0.999$, $\epsilon = 10^{-8}$) and we aimed to minimize the cross-entropy loss, which is given by:

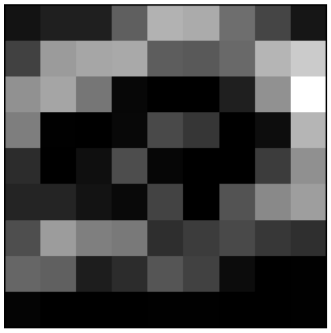
$$\begin{aligned} loss(x, class) &= -log(\frac{exp(x[class])}{\sum_j exp(x[j])}) \\ &= -x[class] + log(\sum_j exp(x[j])) \end{aligned} \tag{2.3}$$

where *class* is the digit to be classified.

To be able to implement the ANN hardware in an efficient manner, we first investigated the effects of image size reduction (downsampling) and quantization of neuron weights. Multiple ANNs with varying image sizes (from 1 pixel/side to 28 pixels/side) and varying weight quantization bits (1 to 8 bits) were created, trained, and tested. To ease the hardware implementation and consecutive implementation steps that will be explained in the next sections, in addition to standard training settings, we constrained the minimum value of neuron weights to be 0. Therefore, all the weights obtained from the training were either 0 or positive numbers. The results of our simulations are presented in Figure 4. During our tests, maximum accuracy achievable from the presented single-layer shallow ANN was 92.95% (for an image size of 26x26 and 8-bits quantization), which is in line with the classification accuracy of single-layer ANNs in the literature [26].

After the successful software implementation, training and testing of the ANN, multiple steps were taken to prepare the design for time-mode hardware implementation. First, image size and number of quantization bits were chosen for the required accuracy. In this implementation, we opted for a 9 pixels/side input image (81 input pixels) in order to i) reduce the energy dissipation without significant loss of accuracy, ii) to have an ANN implementation that is directly comparable to an implementation in the literature [13], and iii) to reduce the transistor level transient simulation time significantly. With input image scaling, the maximum digit classification accuracy reduced from 92.95% to 89.65% (for 8-bit quantization).

After the input image size was chosen, the weights were scaled to the range $[0, 1]$, and were later quantized. From our simulations (Figure 4), 4-bit quantized weights were a good



0.0688	0.1109	0.1141	0.3420	0.6391	0.6135	0.3960	0.2491	0.0794
0.2350	0.5564	0.5936	0.6062	0.3338	0.3202	0.3761	0.6494	0.7286
0.5213	0.5924	0.4227	0.0269	0.0000	0.0020	0.1141	0.5188	0.9153
0.4530	0.0069	0.0028	0.0312	0.2654	0.1935	0.0034	0.0466	0.6476
0.1581	0.0023	0.0542	0.2788	0.0247	0.0005	0.0012	0.2208	0.5170
0.1289	0.1293	0.0697	0.0353	0.2408	0.0028	0.3002	0.4903	0.5672
0.2818	0.5601	0.4566	0.4364	0.1664	0.2205	0.2665	0.1994	0.1696
0.3678	0.3464	0.1063	0.1577	0.3042	0.2379	0.0447	0.0004	0.0044
0.0176	0.0002	0.0014	0.0012	0.0060	0.0053	0.0008	0.0003	0.0045

0.0625	0.0625	0.0625	0.3125	0.6250	0.5625	0.3750	0.1875	0.0625
0.1875	0.5000	0.5625	0.5625	0.3125	0.3125	0.3750	0.6250	0.6875
0.5000	0.5625	0.3750	0.0000	0.0000	0.0000	0.0625	0.5000	0.8750
0.4375	0.0000	0.0000	0.0000	0.2500	0.1875	0.0000	0.0000	0.6250
0.1250	0.0000	0.0000	0.2500	0.0000	0.0000	0.0000	0.1875	0.5000
0.1250	0.1250	0.0625	0.0000	0.1875	0.0000	0.2500	0.4375	0.5625
0.2500	0.5000	0.4375	0.3750	0.1250	0.1875	0.2500	0.1875	0.1250
0.3125	0.3125	0.0625	0.1250	0.2500	0.1875	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Figure 5. Weights of a trained neuron, visual representation and values before and after quantization.

compromise between the expected energy dissipation and accuracy. In the implemented system, which will be explained in the next section, most of the energy is dissipated in the switched capacitances. For a fixed total switched capacitance, the number of quantization bits have negligible effect on the total energy dissipation, and employing a higher number of quantization bits only result in added implementation complexity. However, if smaller capacitance values can be tolerated, i.e., less stringent noise and mismatch considerations in the system, then the smallest number of quantization bits for a given image size, hence the smallest total capacitance values in the implementation, should be chosen as the energy dissipation scales linearly with the number of quantization bits. Furthermore, energy dissipation increases quadratically with the square of image size/side, making the image size a more important parameter for energy reduction. Therefore, the smallest possible image size that satisfies the accuracy requirements should be chosen to minimize the energy dissipation. For the presented ANN implementation, we assumed 9x9 input images, and chose 4-bit quantization weights to represent an average case for the number of quantization bits. The classification accuracy loss due to quantization was minimal, i.e., from 89.65% to 89.35%.

Results of quantization on the weights of the neuron (used to classify handwritten digit 9) are shown in Figure 5. The leftmost figure shows the intensity of the weights, darker pixels representing smaller values. The middle figure shows the floating point scaled weights and the rightmost figure shows the weights after quantization. When these two figures are compared, it is observed that due to quantization, many weights were reduced to *zero*. These *zero* weights have no effect on the weighted sum given in (2.1), therefore can be removed to both simplify the hardware implementation and reduce energy dissipation. Non-zero weights for all the neurons are given in Table 1.

Table 1. Number of non-zero weights after quantization for all the neurons in the designed ANN.

Neuron no.	Non-zero weights
0	63
1	51
2	47
3	39
4	61
5	50
6	54
7	45
8	64
9	53

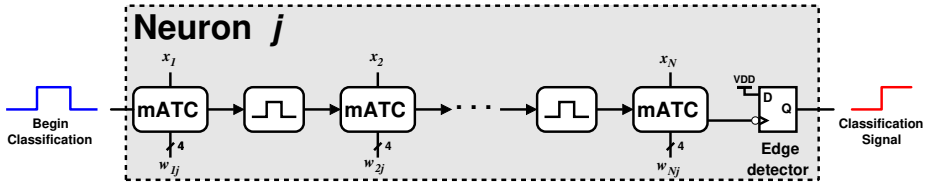


Figure 6. A classifier neuron using time-mode signal processing.

Due to the nature of the designed TMSP circuits which will be explained in the next section, each circuit that realizes the multiply-accumulate (MAC) operation has an inherent non-zero fixed delay. Therefore, not to penalize the neurons which have more non-zero weights and for the correct operation of the designed system, we designed the circuit implementation of the ANN such that each neuron has an equal number of MAC elements, which is equal to the maximum number of non-zero weights given in Table 1, i.e., 64 for neuron 8. Such an implementation allowed us to reduce the number of MAC units for the 9×9 pixel design from 810 to 640, effectively reducing the expected average energy dissipation by 21% by high-level design choices.

3. A Time-Mode MNIST Digit Classifier ANN Implementation

Following the mathematical modeling, training, verification and quantization of the ANN, we applied time-mode operation and TMSP methods to the design of a digit classification ANN in a standard 0.18 μm IC process. Each neuron defined by (2.1) is mapped to a TMSP implementation, as shown in Figure 6. As in (2.1), a chain of multiplying analog-to-time converters (mATC) converts a voltage input value into a pulse whose width is proportional to both the input signal value and the assigned weight. The signal propagates through the chain of mATCs and fixed-width pulse generators (FWPGs). FWPGs, represented by the pulse blocks in Figure 6, are required to be able to trigger the next mATC in the chain with the falling edge of the previous mATC pulse. The structures and operation principles of both the mATC and the negative-edge triggered fixed-width pulse generator are explained in the following paragraphs. In this specific implementation, we created a chain of 64 mATCs for each neuron. Due to the resulting zero weights after quantization, not all the pixels are connected to each neuron, further simplifying the hardware implementation and future on-chip routing.

The operation of the TM ANN neuron is as follows: Once the neuron has been triggered with the *Begin Classification* signal, the chain of mATCs and FWPGs operate sequentially to accumulate the delay information from each mATC, each of which represents the weighted input pixel data. As explained in the previous section, the ANN has been trained with a Softmin activation function, meaning that the neuron with the smallest weighted sum output value, i.e., in TMSP terms, the fastest response (earliest falling edge at the output of the last (N-th) mATC), will get to classify the input image first. Therefore, we placed negative-edge triggered flip-flops at the output of the neurons to capture the final falling edge of the signal generated by the chain of mATCs. This "faster response wins" approach directly mimics the Softmin function explained in the previous section and is also similar to how some biological neural networks which are trained repeatedly behave.

During the design of the TM ANN, we employed a modified version of the basic monostable multivibrator (MSMV) [27] to work as an mATC in the system, as shown in Figure 7. In this implementation, a pMOS transistor (M1) acts as a variable resistor whose resistance is modulated by the current input voltage signal. When the MSMV is triggered by an input pulse, nodes **n1** and **n2** are pulled to logic-low and M1 starts charging node **n2**. The gate of M1 is driven by the input signal that is to be converted into time, and sampling is realized by modulating the instantaneous resistance of M1. Thus, the RC time constant of the multivibrator is modulated as well, resulting in a pulse whose width is proportional to the amplitude of the input signal. The pulse width

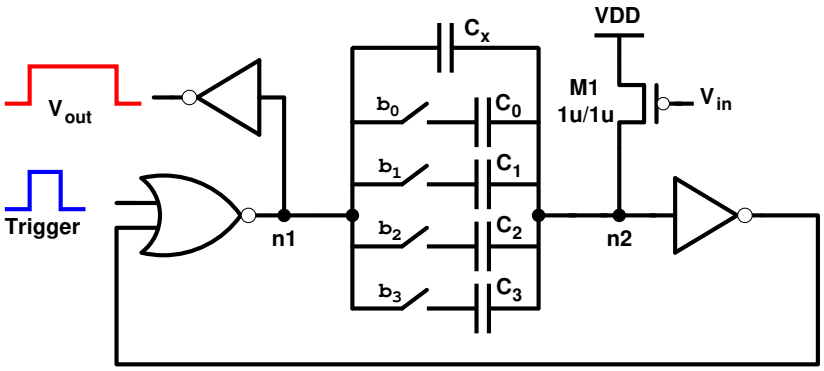


Figure 7. Monostable multivibrator based multiplying ATC with time linearizing capacitor C_x.

generated by the ATC is given in [28] by:

$$T = C(R + R_{on}) \ln \left[\frac{R}{R + R_{on}} \frac{V_{DD}}{V_{DD} - V_{th}} \right], \tag{3.1}$$

where R is the average resistance of the pMOS transistor during pulse generation, R_{on} the resistance of the NOR gate, and V_{th} the switching threshold of the inverter. Assuming $R_{on} << R$ and $V_{th} = V_{DD}/2$, (3.1) is simplified to $T = 0.69RC$. Furthermore, this mATC implementation has an inherent *timeout* feature and will always generate a pulse event at node **n1** regardless of the input signal value at V_{in}, avoiding stalling of the chain. Transistor M1 was made bigger than the minimum values required for correct operation to mitigate process variation effects. The ATC given in [28] was modified with the inclusion of extra switchable capacitors C_0-C_3 to allow the ATC to realize time-multiplication operation. The capacitors C_0-C_3 are increasing in a binary weighted fashion, C_0 being the unit least-significant bit (LSB) capacitance, and $C_3 = 8 \cdot C_0$ being the most-significant bit (MSB) capacitance. The unit capacitor was sized such that, for the smallest multiplication coefficient, i.e., `0001`, the mATC still generates a pulse response that is proportional to the input signal value. The switches were implemented as transmission gates using minimum size (0.22μm/0.18μm) MOS transistors.

In the first iteration of the design, the minimum unit capacitor that satisfies this requirement was found to be 20 fF. In this iteration, we used only switchable capacitors as the charged capacitor to reduce the total switched capacitance, hence the total energy dissipation. However, during our transistor level simulations, we saw that due to the parasitic capacitances at node **n2** and the non-idealities of the switches, the pulse-width ratio between the successive weights degraded, especially for the smaller values, i.e., for `0001` and `0010`. Therefore, we placed a fixed *time linearizing* 10 fF capacitor C_x in parallel to the switched capacitors. Addition of C_x also allowed us to reduce the value of the unit switched capacitor from 20 fF to 10 fF, as for the smallest weight setting, the charged capacitance at **n2** is still 20 fF.

Transistor level simulations using the HSPICE simulator were run to characterize the mATC. Simulations were run for a supply voltage of 0.6 V VDD, while sweeping the input signal voltage from 300 mV to 400 mV, to represent the expected input signal values from an imager. In all the transistor level simulations, the black and white pixels are represented by 300 mV and 400 mV input voltage values to the mATCs, respectively.

The advantages of the placement of C_x in the second iteration of the mATC design are shown in Figure 8. The range of pulses generated by both versions of the mATC for different weights as well as their mean are presented in the figure. There are multiple points that should be noted from transistor level simulation results: i) slope of the time response of the mATC has been reduced, effectively making the mean pulse-width values more fitting to a binary progression, hence the name time-linearizing (TL), ii) due to the better fitting of the mean to binary progression, the

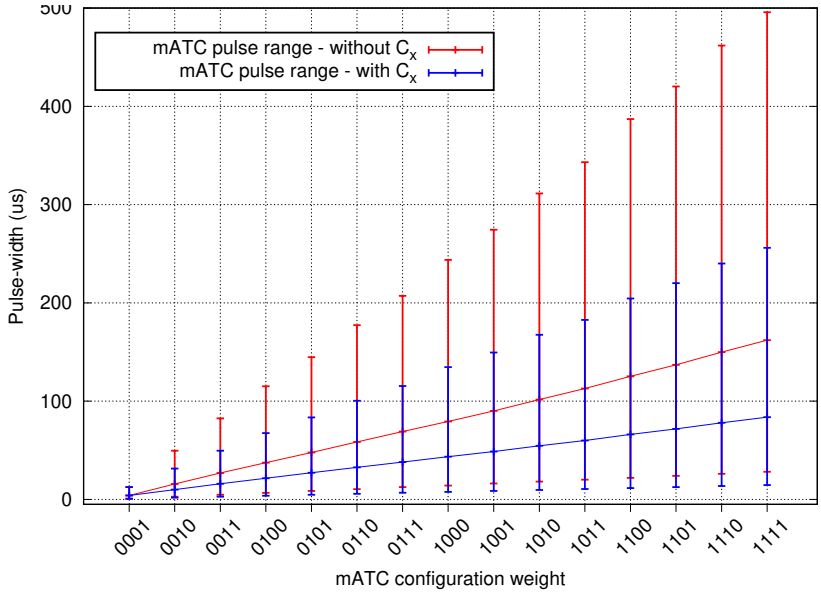


Figure 8. Pulse-width ranges generated by the mATC for different configuration weights.

error between the multiplication steps has been reduced (the root mean squared error (RMSE) is reduced from 6.61% to 1.59%, see Table 2 for more details), and iii) due to the reduced total capacitance, the system response is faster (average pulse width is reduced from 81.16μs to 43.72μs) and the average energy dissipation is reduced (from 254 fJ to 157 fJ).

A negative edge triggered FWPG, shown in Figure 9, is used between the mATC blocks as we require the triggering of the next mATC in the chain to occur during the falling edge of the pulse generated by the previous mATC. By triggering the next mATC with the falling edge of the previous mATC output, time addition operation is realized. In this implementation, we used a FWPG which generates pulses with a pulse-width of 50 ns. This minimum value of the pulse-width can be chosen to be any value that satisfies the following requirements during triggering:

Table 2. mATC expected multiplication weight ratios and % errors for different designs.

mATC multiplication weight ratio	Expected ratio	% error - base mATC	% error - TL mATC
w_2/w_1	2.00	91.09	20.91
w_3/w_2	1.50	15.03	7.17
w_4/w_3	1.33	4.27	0.22
w_5/w_4	1.25	2.87	1.62
w_6/w_5	1.20	1.96	0.41
w_7/w_6	1.17	1.50	0.41
w_8/w_7	1.14	-0.24	-0.87
w_9/w_8	1.12	1.11	0.26
w_{10}/w_9	1.11	1.36	0.45
w_{11}/w_{10}	1.10	1.08	0.27
w_{12}/w_{11}	1.09	0.99	0.45
w_{13}/w_{12}	1.08	1.07	0.37
w_{14}/w_{13}	1.08	1.34	0.74
w_{15}/w_{14}	1.07	1.08	0.48

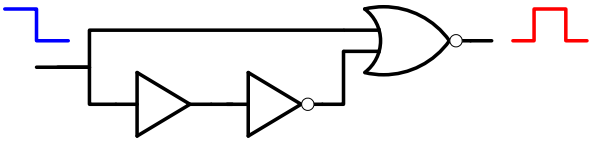


Figure 9. Negative-edge triggered fixed-width pulse generator.

i) both nodes **n1** and **n2** are completely driven to ground during the pulse, and ii) other input of the NOR gate is completely driven to VDD with sufficient timing margin to account for process mismatch before the output of the FWPG goes low. The maximum value of the pulse-width of the FWPG is limited by the minimum pulse value that is generated by the mATC, i.e., 1.94 μs for a `0001` input. During our simulations, same pulse-width was also used for the *Begin Classification* signal.

4. Simulation Results

After characterization and verification of the sub-blocks of the hardware ANN, extensive transistor level SPICE simulations using HSPICE simulator were run to verify the correct time-mode operation of the designed system. As in the characterization of the sub-blocks, a supply of 0.6 V is used. Separate testbenches were programmatically created to simulate 100 samples from the test dataset and transient simulations were run. Results of one such simulation run for a classification of digit 2 is shown in Figure 10. As it can be seen, the correct classifier neuron generates a faster output response than the other neurons, successfully classifying the input digit. For this specific case, the fastest neuron, i.e., neuron 2, responded 62.1μs faster than the next fastest neuron.

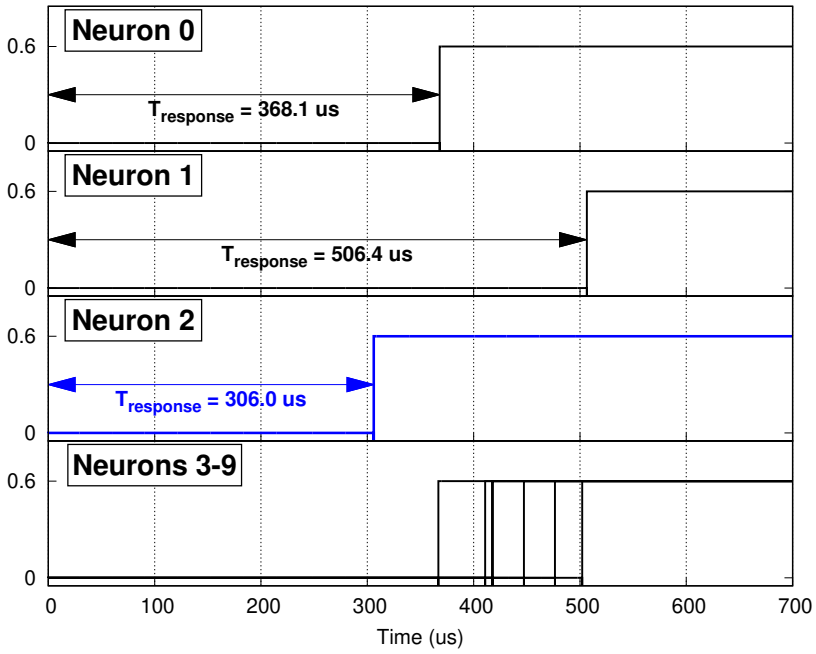


Figure 10. Transistor level transient simulation of an example correct classification of a handwritten digit 2 by neuron 2. The figure shows the output signals of the neurons with timing information.

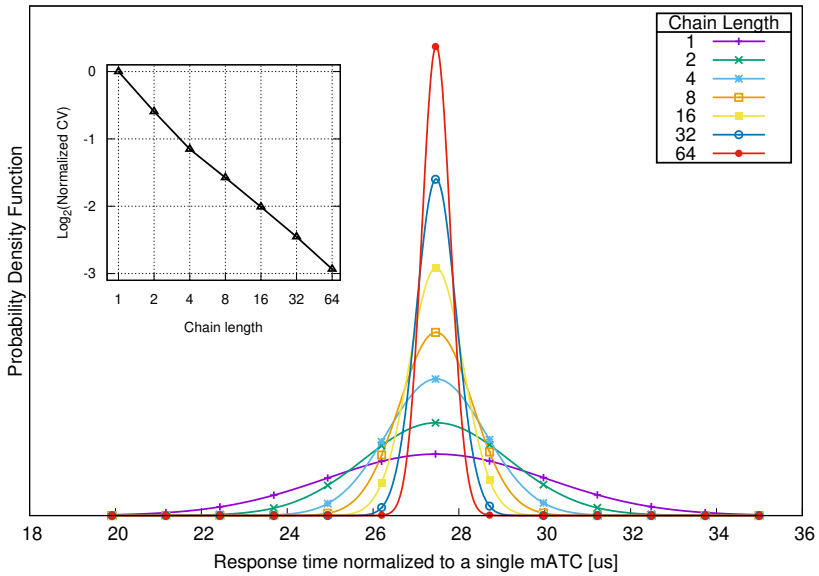


Figure 11. 100-point Monte-Carlo simulations showing the improvement in the coefficient of variation with increasing number of mATCs inside the neuron.

The average energy dissipation per classification while working at 0.6 V VDD is 65.74 pJ. The average classification response time for the test dataset is 421.8μs, resulting in 2.37 k classifications per second for a classification accuracy of 88%. As the focus of the present study is on an energy efficient design with low dissipation rather than state-of-the-art classification accuracy, the accuracy of 88%, which is typical in 1-layer neural networks [26], is acceptable at the current stage. In the meantime, the classification accuracy is still significantly higher than the random guess for MNIST dataset.

We also investigated the effects of process mismatch on the performance of, first, a chain of mATCs, and later, on the ANN. We first simulated a chain of mATCs with varying number of elements for the effects of local mismatch. For these simulations, to represent an average case of operation, all the analog input voltages to the mATCs and the multiplication coefficients were set to 350 mV and `1000`, respectively. 100-point Monte Carlo simulations were run, and the results are presented in Figure 11. The figure shows the curve fits of the normalized probability density functions over a varying number of elements in an mATC chain and the decrease in the coefficient of variation ($CV = \sigma/\mu$) with increasing number of elements in the chain. The improvement in CV is $\sqrt{2}$ for every doubling of the number of mATC elements in the chain.

As it is apparent from the mATC chain mismatch simulations, increasing the number of elements in the chain reduces the relative variability of the ANN. Even though as the implemented ANN is trained off-line and there is no provision and straight-forward way to address variability during training, reliability issues due to process mismatch may be addressed in two ways: i) Algorithmically testing each neuron for the variability of the elements by applying multiple analog input and digital control combinations and extracting the linear transfer curve, and ii) by increasing the number of mATCs in the chain to average out and reduce the effects of variation, as shown in Figure 11.

To test the performance of the ANN for process mismatch, for each of the 100 image samples we used to simulate and characterize the system, we ran 100 point Monte-Carlo mismatch simulations (100x100 transient simulations in total) and the average standard deviation in the neuron response due to process mismatch time was 9.2μs. Simulation results of such a simulation run for the classification of handwritten digit of 3 is presented in Figure 12 as an example. The

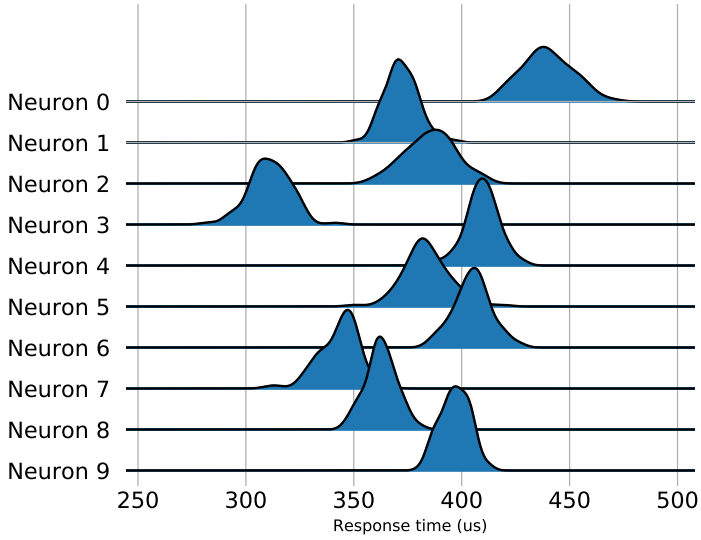


Figure 12. 100 point Monte-Carlo simulation (classification of a handwritten digit 3) response time variation of the neurons of the designed ANN due to process mismatch.

figure shows the response time variation distribution of each neuron in the designed ANN due to process mismatch.

For a misclassification to occur due to mismatch, the second fastest neuron should respond faster than the fastest neuron of the nominal conditions. For the case shown in Figure 12, this is possible between neurons 3 and 7. This probability can be modeled as the half of the area of overlap of two Gaussian distributions with the same standard deviation and differing means (Figure 13), and the intersection point of the distributions depends on the distance between the means. For mean differences less than 1.2σ, we saw that the errors due to the training (88% accuracy) occurred. For mean differences greater than 1.2σ, we calculated the added misclassification probability for each neuron and found the total added possibility of error due to process mismatch to be 1.17%, reducing the expected *minimum* accuracy to 86.63%.

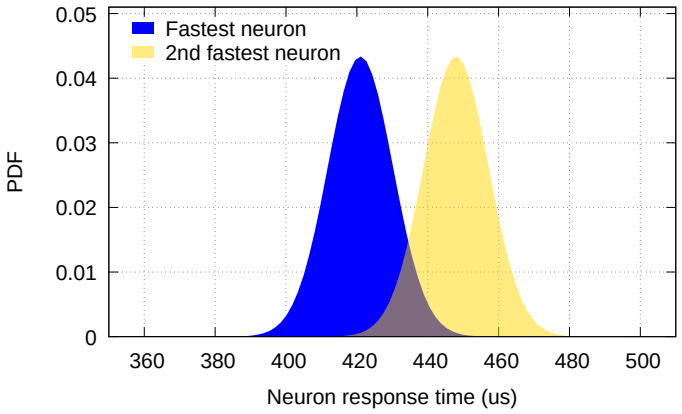


Figure 13. Response variation of 2 competing neurons due to process mismatch. Half of the value of the green shaded area represents the probability of misclassification.

Table 3. Comparison of the implemented TM ANN with the implementations in the literature.

	SRAM Classifier [13]	This work
Technology [nm]	130	180
Supply voltage [V]	1.2	0.6
Classification accuracy [%]	90	88
Classification Speed [Hz]	$50 \cdot 10^6$	2370
Analog-to-digital conversion energy included	No	Yes
Energy dissipation [pJ]	630	66

When compared to the state-of-the-art hardware ANN implementations, the design presented in this work compares favorably in terms of reduced energy dissipation, which is the main aim of this design exercise. A comparison of results with a recent and directly comparable hardware 9×9 pixel MNIST classification ANN in [13] is given in Table 3. When both implementations are compared, even though the presented ANN is designed using an older technology, i.e., 0.18μm process, compares favorably in terms of energy dissipation. One metric where the design in [13] is performing better than the presented implementation is the classification speed. However, due to the design constraints in [13], operating voltage cannot be lowered further, hence, energy dissipation, which is proportional to the square of supply voltage in digital circuits, cannot be further reduced. Furthermore, it is expected that our implementation will achieve much better average operating speed and energy dissipation numbers when this design is migrated to more advanced technologies. The energy dissipation per classification is reduced by a factor of **9.58x**, from $630\,$pJ down to $65.74\,$pJ when compared to [13]. It should also be noted that the ANN implementation presented in this paper works with analog signal inputs, without requiring the input data to be converted to digital for further processing. If analog-to-digital conversion energy cost per image is added to the classification energy numbers reported in [13] in Table 3, presented ANN implementation is even more energy efficient.

Extending the single-layer ANN presented in this study to a multi-layer version is an on-going work. However, from our preliminary results, it has been observed that, once a value/variable is converted to a time-mode signal, in order to operate in the most energy efficient way, processing should continue in time-mode without conversion between the time-mode and analog/digital domains. For example, an asynchronous time-to-digital converter (TDC) in $0.18\,\mu$m process dissipates 1.48 pJ [19], and a similar TDC in a 65 nm process dissipates 0.97 pJ [29] per conversion. When compared to the average energy dissipation of each neuron (6.6 pJ), it can be observed that conversion between different operating domains incur energy dissipation overhead values which are comparable to the energy dissipation of the data processing circuitry.

5. Conclusions

This paper presents the hardware design and the simulation results of a time-mode, single-layer artificial neural network (ANN) with Softmin activation function for handwritten digit classification. Time-mode signal processing techniques have been applied for accumulating weighted image signal values using energy-efficient time-mode circuitry. Optimization steps for both system level and hardware level design are given. The system was designed and simulated in a standard $0.18\,\mu$m process and operates from a supply voltage of 0.6 V. By applying the presented design guidelines, an energy-optimal 9×9 handwritten digit image classification ANN with 4-bit quantized weights was designed. The energy dissipation of the design for each classification is $65.74\,$pJ while operating at a speed of $2.37\,$k classifications per second, with a classification accuracy of 88%.

6. Enunciations

Authors' Contributions. OCA conceived, designed, simulated and verified the transistor level, time-mode ANN implementation. OCA and JM created and optimized Python level ANN for time-mode implementation. Both authors drafted, read, and approved the manuscript.

Competing Interests. The author(s) declare that they have no competing interests.

Funding. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 752819 for the MSCA IF Project ATiNaRI.

Acknowledgements. The authors thank the anonymous reviewers for their constructive comments and helpful suggestions.

References

1. I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.
2. K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.
3. D. Cireşan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," *arXiv preprint arXiv:1202.2745*, 2012.
4. J. Ba, V. Mnih, and K. Kavukcuoglu, "Multiple object recognition with visual attention," *arXiv preprint arXiv:1412.7755*, 2014.
5. L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8599–8603.
6. S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2013.
7. M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3367–3375.
8. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.
9. D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.
10. J. Khan, J. S. Wei, M. Ringner, L. H. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C. R. Antonescu, C. Peterson *et al.*, "Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks," *Nature medicine*, vol. 7, no. 6, p. 673, 2001.
11. Q. K. Al-Shayea, "Artificial neural networks in medical diagnosis," *International Journal of Computer Science Issues*, vol. 8, no. 2, pp. 150–154, 2011.
12. S. Kodali, P. Hansen, N. Mulholland, P. Whatmough, D. Brooks, and G.-Y. Wei, "Applications of deep neural networks for ultra low power iot," in *2017 IEEE International Conference on Computer Design (ICCD)*. IEEE, 2017, pp. 589–592.
13. J. Zhang, Z. Wang, and N. Verma, "In-memory computation of a machine-learning classifier in a standard 6t sram array," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 4, pp. 915–924, 2017.
14. Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.
15. J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "Unpu: An energy-efficient deep neural network accelerator with fully variable weight bit precision," *IEEE Journal of Solid-State Circuits*, 2018.
16. D. Bankman, L. Yang, B. Moons, M. Verhelst, and B. Murmann, "An always-on 3.8 μj/86% cifar-10 mixed-signal binary cnn processor with all memory on chip in 28nm cmos," in *Solid-State Circuits Conference-(ISSCC), 2018 IEEE International*. IEEE, 2018, pp. 222–224.

17. F. Yuan, "CMOS time-to-digital converters for mixed-mode signal processing," *The Journal of Engineering*, January 2014. [Online]. Available: http://digital-library.theiet.org/content/journals/10.1049/joe.2014.0044
18. Z. Chen and J. Gu, "Analysis and design of energy efficient time domain signal processing," in *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*. ACM, 2016, pp. 100–105.
19. O. C. Akgun, M. Mangia, F. Pareschi, R. Rovatti, G. Setti, and W. A. Serdijn, "An Energy-Efficient Multi-Sensor Compressed Sensing System Employing Time-Mode Signal Processing Techniques," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2019, pp. 1–5.
20. Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.
21. A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.
22. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
23. J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.
24. T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.
25. S. Ruder, "An overview of gradient descent optimization algorithms," 2016.
26. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
27. O. C. Akgun, F. K. Gurkaynak, and Y. Leblebici, "A Current Sensing Completion Detection Method for Asynchronous Pipelines Operating in the Sub-threshold Regime," *International Journal of Circuit Theory and Applications*, vol. 37, pp. 203–220, 2009.
28. A. S. Sedra and K. C. Smith, *Microelectronic Circuits*, 4th ed. Oxford University Press, 1998.
29. O. C. Akgun, "An asynchronous pipelined time-to-digital converter using time-domain subtraction," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5.