Programmable Neuromorphic Circuit based on Printed Electrolyte-Gated Transistors

Dennis D. Weller*
Karlsruhe Institute of Technology
dennis.weller@kit.edu

Michael Hefenbrock*
Karlsruhe Institute of Technology
michael.hefenbrock@kit.edu

Mehdi B. Tahoori
Karlsruhe Institute of Technology
mehdi.tahoori@kit.edu

Jasmin Aghassi-Hagmann
Offenburg University of Applied Sciences
jasmin.aghassi-hagmann@hs-offenburg.de

Michael Beigl
Karlsruhe Institute of Technology
michael.beigl@kit.edu

Abstract— Neuromorphic computing systems have demonstrated many advantages for popular classification problems with significantly less computational resources. We present in this paper the design, fabrication and training of a programmable neuromorphic circuit, which is based on printed electrolyte-gated field-effect transistor (EGFET). Based on printable neuron architecture involving several resistors and one transistor, the proposed circuit can realize multiply-add and activation functions. The functionality of the circuit, i.e. the weights of the neural network, can be set during a post-fabrication step in form of printing resistors to the crossbar. Besides the fabrication of a programmable neuron, we also provide a learning algorithm, tailored to the requirements of the technology and the proposed programmable neuron design, which is verified through simulations. The proposed neuromorphic circuit operates at 5V and occupies 385mm^2 of area.

I. Introduction

In recent years, neuromorphic computing systems (NCS) have shown to be very potent solutions to efficiently solve neural network (NN) tasks, such as classification or regression problems with reduced computation time [1], high energy-efficiency on hardware fabricated in compact volume and with inherent fault-tolerance [2]. These attractive features motivated research groups to implement NCSs in emerging technologies. On the other hand, Printed Electronics (PE) has gained a lot of attention recently. In comparison to traditional silicon-based VLSI, PE can potentially decrease the fabrication costs of electronic systems using large-area printing and provides additional properties for future applications based on flexible, green and low-voltage devices.

One major benefit for realizing NCS in PE is the mitigation of high device latencies of printed designs by taking advantage of the parallel computing capability of the NCS. The analog computing paradigm of NCS can also be used to replace digital circuits to obtain smaller hardware footprints and further lower device variations by reducing transistor count, typically existing in boolean-logic-based designs. Moreover, the crossbar architectures can easily be customized according to demand during a post-fabrication step by means of weight-adjustment through resistor printing.

In this paper, we propose new design concepts and tools such as a mapping and learning algorithm to deploy NN computations on a programmable non-spiking NCS with a customizable crossbar array and an activation function using inkjet printed electrolyte-gated inorganic transistor technology. *To the best of our knowledge, this is the first time a NCS is fabricated and demonstrated using printing technology.* To be precise, we make the following contributions:

1) We design, fabricate and characterize a programmable neuron, which consists of a crossbar-array with post-fabrication of weights and a printed activation function circuit involving only one transistor (EGFET).
2) We have developed a training algorithm tailored for the requirements of the technology and printed neuron circuitry.
3) We verified the proposed programmable NCS and its learning algorithm simulation-based.

This work might encourages other PE research groups to consider NCS-based implementations in their design flow. To this end, we provide a reproducible concept for printed neurons with a set of learning rules to solve classification problems by the NCS. This is especially valuable as existing digital-based printed designs can be substituted to improve circuit characteristics or even broaden the scope of potential applications.

The rest of this paper is structured as follows: In Section II, we provide background of this work. In Section III, the proposed printed neuron design is presented and in Section IV, we describe the fabrication of the printed neuron circuitry. Section V introduces the developed learning algorithm to solve a specific classification tasks. Finally, Section VI concludes the paper.

II. Preliminary

A. Printed Electronics and Electrolyte-gated inorganic transistors

Printed Electronics (PE) offer solutions beyond the capabilities of silicon-based VLSI technologies, such as flexible, large-area and low-cost fabrication. They can be classified into different groups, according to the utilized processes and materials. Organic-based electronics for instance are widely used, which require however high supply voltages ranging up to 100V [3] and can thus limit the deployment in embedded applications. On the other side, inorganic n-type electrolyte-gated field-effect transistors (EGFET) where developed, with sub 1V operation region [4].

In our work, we deploy inkjet printing, which offers a digital, customizable post-processing step that can be used to

*Both authors have same contribution to this work

978-1-7281-4123-7/20/$31.00

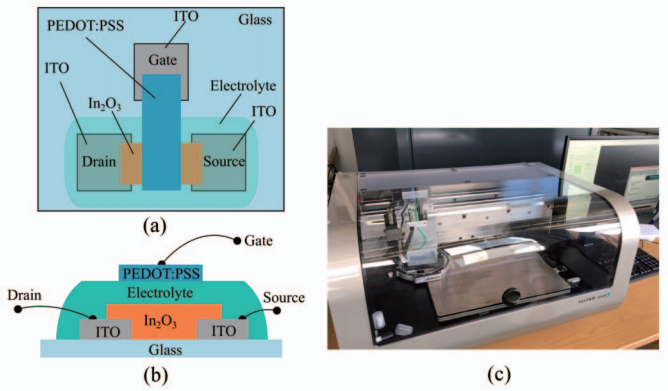


Fig. 1: Printed Electronics technology (a) top- and (b) side-view of the electrolyte-gated field effect transistor (c) Fujifilm Dimatix Materials Inkjet Printer DMP-2850

easily tailor the functionality of the printed NCS. In contrast, in screen printing or roll-to-roll printing, a master plate has to be fabricated as part of the replication process. Inkjet printing on the other side enables on-demand printing and allows circuit customization and tuning.

Inkjet printing as a fully additive process was proven to be very efficient to implement digital circuits such as logic gates, memory elements or physical unclonable functions, to name a few [4, 5, 6]. The used EGFETs in these designs are build from the following functional inks: In_2O_3 as the semiconducting channel material, an electrolyte as a substitute for a dielectric layer and PEDOT:PSS as the conductive material used in resistors and the top gate electrode [4]. In Fig. 1, the transistor stack as well as our inkjet-printer are illustrated. Using electrolytes offers the advantage to drastically reduce the supply voltage due to its high gate-capacitance.

As EGFET-based circuits are part of an emerging and less mature technology, still several challenges remain unsolved. Due to inherent non-determinism in droplet printing as well as the strong impact of interface quality between the different thin film layers, high process variabilities exist and thus designs are preferred with a low number of devices. Therefore, EGFET-based designs aim at low-complexity implementations with reduced number of transistors. Furthermore, due to the absence of p-type transistors, mapping of existing designs to the limited NMOS-logic sometimes is difficult. NCS with resistor-based crossbar architectures are thus a suitable solution to build circuits that do not demand many transistors. Moreover, NCS does not require high-performance devices and can achieve reasonable performance even with high device latencies, which is another characteristic of PE.

B. Related Work

Several crossbar architectures for NCS have been presented in recent years. The authors of [7] reported on a memristor-based crossbar architecture in combination with summing amplifiers to realize a Brain-State-in-a-Box (BSB) model for optical character- and image processing. Based on a similar hardware, [8] provided an automation framework for automatic deployment of NNs to memristor-based crossbars. Also on the level of algorithms, most recently, [9] described how quantization and bias-tuning techniques can be used to compensate for NN accuracy losses.

However, these silicon-based architectures and concepts can not be applied to existing printed electronic technologies due to fundamental differences in material properties and design complexity. Nevertheless, organic-based crossbar-architectures were introduced in the past [10]. The focus of this work was however more on device characterization, not on developing circuits for NCSs. In contrast, [11] reported on a low-complexity design for both memristor-based crossbar and activation function based on organic p-type transistors. The solution proposed in [11] is however not programmable by printing as it is the case in all other prior published works.

III. Programmable Neuron Design

Our proposed inkjet-printed circuits are build from n-type electrolyte-gated transistors (EGFET) and printable resistors. Although the lack of p-type transistors in this technology is a limitation, we can still manage to realize efficient neurons for neural networks based on our NMOS-logic. After offline training for classification tasks, the neural network is customized by printing the corresponding weights to the crossbar array. The schematic of the neuron is depicted in Fig. 2. It consists of two components required to build NN architectures: the multiply-add circuit and the activation function. All design considerations regarding the neuron implementation will be described in the following.

A. Multiply-Add

As a digital implementation with boolean logic would require very complex designs, the weighted-sum operation can easier be realized by a crossbar architecture. In this low-complexity solution, the input voltages are converted into currents, inversely proportional to the resistances on the crossbar-points, and are instantly summed up according to Kirchhoff's rule. The novelty of this approach is, that the resistors of the crossbar array can be customized by inkjet-printing during post-fabrication and tuned in their value by printing. Therefore, PEDOT:PSS ink can be printed in different geometries to the crossbar-points to realize different resistances.

The output voltage of the crossbar is determined by the voltage drop across R_{base} and can be computed as:

$$V_x = \frac{R_{base}\left(\sum_{i=1}^{P}\frac{V_i}{R_i}\right)}{1 + R_{base}\left(\sum_{i=1}^{P}\frac{1}{R_i}\right)} \tag{1}$$

For $R_{base} \gg R_i$, Equ. 1 can be approximated, according to Equ. 2:

$$V_x = \frac{\sum_{i=1}^{P}\frac{V_i}{R_i}}{\frac{1}{R_{base}} + \left(\sum_{i=1}^{P}\frac{1}{R_i}\right)} \approx \frac{\sum_{i=1}^{P}\frac{V_i}{R_i}}{\sum_{i=1}^{P}\frac{1}{R_i}} = \sum_{i=1}^{P} V_i\ w_i \tag{2}$$

As a result, the synaptic weights can be written in the form of:

$$w_i = \frac{\frac{1}{R_i}}{\sum_{j=1}^{P} \frac{1}{R_j}} \tag{3}$$

From Equ. 3 it is obvious that the w_i are bounded: $w_i \in [0, 1]$. In addition, by summing up all w_i, we obtain the constraint: $\sum_{i=1}^{P} w_i = 1$.

It is important to mention that there exist another approximation of Equ. 1 by setting $R_{base} \ll R_i$, which was used by [11]. Using this approximation, the constraint $\sum_{i=1}^{P} w_i = 1$ is not existing, however, this leads to an upper bound on the weights, which is much smaller than 1, and thus leads to a high voltage drop across each node causing signal losses towards the NCS output.

B. Activation Function

Activation functions for neural networks introduce non-linear behavior to the neuron computation. As illustrated in Fig. 2, the printable activation function design consists of an n-type EGFET with two resistances at the gate. They realize a voltage-divider, which turns the transistor on for positive polarity, and off for negative polarity. To this end, resistor sizing of the voltage divider has to fulfill: $R_L \gg R_H$. Dependent on the crossbar output voltage V_x, the input voltage of the activation function is positive or negative, and either high positive drain currents or small negative drain currents flow through the pull-down resistor R_{out}, which generates the output voltage V_{out} of the neuron. As we later show in Section IV, the resulting activation function represents a piece-wise linear unit. In order to guarantee that the multiply-add-circuit is not shortcut by the activation function circuit, $R_{out} \gg R_{base}$ has to be satisfied. In a similar manner, the series resistance of the voltage divider has to be higher than the ON-resistance of the EGFET: $R_L \gg R_{ON}$. Otherwise the transistor operation point would be shifted or the transistor will be shortcut by the voltage divider, respectively.

C. Biasing

The constraint $\sum_{i=1}^{P} w_i = 1$ can create a significant restriction for the training algorithm, forcing the weight vector to lie on a hyperplane. To relax this condition, we can add an additional "dummy" input with a weight of $w_b = 1 - \sum_{i=1}^{P} w_i$ and an applied input voltage V_{bias} of $0V$ (see Fig. 2). In this case, the constraint changes to $\sum_{i=1}^{P} w_i + w_b = 1 \implies \sum_{i=1}^{P} w_i < 1$. Alternatively, we can also use the "dummy" node to bias the neuron by b, in this case the constant applied voltage must be equal to $V_{bias} = \frac{b}{w_b}$.

The input of the neuron $x_i = V_i$ and the bias b can only be chosen continuously between -1V and 1V. Higher absolute voltages would destroy the electrolyte of the EGFET.

Furthermore, to also limit V_{bias} and hence the maximum required supply voltage, we set $w_b \geq w_b^{min} = 0.2$. In case of a 2-input neuron with $w_1 + w_2 = 0.8 \wedge w_b = 0.2$ and $b = 1$ we have $V_{bias} = \frac{b}{w_b} = \frac{1}{0.2} = 5V$, where $5V$ is the maximum supply voltage. For an unbiased neuron we set $V_{bias} = 0V$.

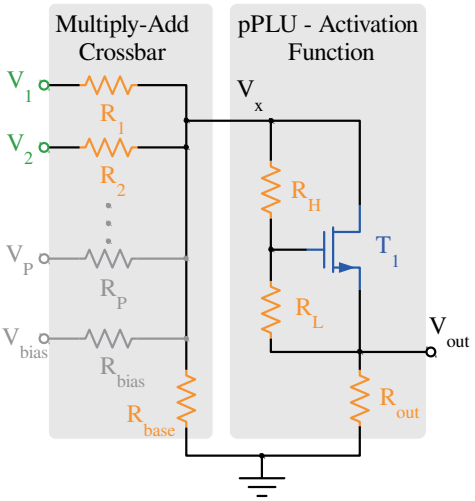


Fig. 2: Neuron schematic - realisation of multiply-add with crossbar array, implementation of activation function with EGFET (T_1), voltage divider (R_H, R_L) at the transistor gate and pull-down resistor (R_{out}). The number of inputs can easily be scaled by adding further input resistances R_i (grey).

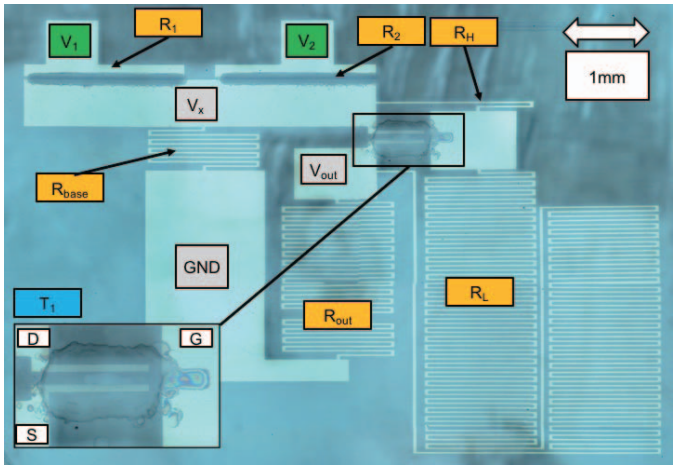


Fig. 3: Photo of inkjet-printed 2-input neuron with printed weights (R_i) and EGFET (T_1), annotated with contacts for drain (D), source (S) and gate (G). Layout was derived from the schematic depicted in Fig. 2 (without grey resistors).

IV. Programmable Neuron Fabrication and Characterization

A. Fabrication of 2-input neuron

We fabricated a two-input neuron ($P = 2$) according to the design depicted in Fig. 2 and based on the inkjet-printing technology introduced in Section II. A 20x20mm ITO-sputtered glass substrate was structured using laser ablation in order to obtain the passive conductive tracks. Afterwards In_2O_3 for the semiconductor channel material of the EGFET was inkjet-printed and annealed at 400°C. Subsequently, the electrolyte was printed covering the channel, source and drain electrodes, respectively (see Fig 1). Finally, PEDOT:PSS was printed for both, the top-gate contact of the EGFET, as well as the

TABLE I: Design parameters for 2-input neuron fabrication

$R1$	$R2$	R_{base}	R_{out}	R_D	R_H	W/L of T1
$1k\Omega$	$1k\Omega$	$30k\Omega$	$300k\Omega$	$500k\Omega$	$10k\Omega$	$200\mu m/80\mu m$

synaptic weights. A microscope photo of the printed circuit is illustrated in Fig. 3. The values of all design parameters and sizing of the transistor are depicted in Table I.

R_{base}, R_{out}, R_D and R_H were sized in accordance to the discussion in Section III. R_{on} and R_{off} resistances of the EGFET for a geometry of $W/L = 200\mu m/80\mu m$ are typically $5k\Omega$ or $2M\Omega$, respectively.

B. Measurements of 2-input neuron

In total three measurements were conducted and the outcome of those are shown in Fig. 4. The first two measurements were performed on isolated circuits, namely one separate multiply-add-circuit and a separate activation function circuit. In the transient measurement of Fig. 4a, the correct operation of a weighted-sum with $w_1 = w_2 = 0.5$ can be observed. The output voltage is pulled up to 1V when both inputs are 1V or pulled down to -1V for inputs of -1V, respectively. For the two other input voltage combinations, the currents at the crossbar cancel out and the output voltage V_x reaches nearly 0V. Regarding the activation function circuit (Fig. 4b), a printed piece-wise linear unit is obtained, which we call in the following pPLU. Figure 4c shows the transient measurement of the neuron response (multiply-add + pPLU), which was fabricated as seen in Fig. 3. As expected, negative input voltage combinations are suppressed due to the pPLU. The frequency of the square pulses V_1 and V_2 was increased to investigate the latency of this device and thus the maximum operating frequency.

The latency of the neuron was about 1.6ms, giving a maximum operating frequency of 625Hz. Output signals could still be differentiated for a maximum voltage swing at the input of -0.7V to 0.7V. The area of the neuron is about $54mm^2$, which can however be further optimized in the future by replacing the ITO passive structure with low-conductive printed PEDOT:PSS.

V. Training Printed Neural Networks

As described before, several constraints and restrictions from the technology and the proposed neuron design need to be factored in for training the proposed programmable and printed NCS for a classification task. In the following, we present a training algorithm as well as an appropriate loss function to deal with the given constraints presented in Section IV III-C, i.e.

- $w^{(n)} \in [0, 1-w_b^{min}]^{n_P} \subset \mathbb{R}^{n_P}$, $\sum_{i=1}^{n_P} w_i^{(n)} \leq 1-w_b^{min}$
- $b^{(n)} \in [-1, 1] \subset \mathbb{R}$.

We use the superscript (n) to distinguish between the neurons in the NN, $w_3^{(2)}$ is for instance the 3rd weight of neuron 2. In addition, we summarize the weights in vector notation, e.g. the parameter $w^{(n)}$ denotes the weight vector of the neuron n with n_P inputs.

A. The Loss Function

An important part of training neural networks is the loss function, where typical choices are e.g. mean squared error (MSE) or cross-entropy (CE) [12]. In both cases, for some data $\mathcal{D} = \{(x_j, y_j)\}_{j=0}^{m}$, the loss function measures a distance of the output of the network $f_\theta(x_j)$ for a training instance x_j to the true label value y_j (usually $y_j \in \{0, 1\}$). The parameter θ thereby denotes all parameters of the network, i.e. weights and biases of all neurons.

However, since the parameters $\theta^{(n)}$ of a printed neuron n are bounded i.e. $w_i^{(n)} \in [0, 1-w_b^{min}]$, the range of the neural network output $f_\theta(x)$ is also bounded. Additionally, the signal x is likely to shrink through each computation due to the multiplication with values smaller than one, i.e. by weights or through the activation function, which reflects the voltage drop. This phenomenon leads to all inputs signals becoming increasingly similar through each mapping (multiply-add and activation), and cannot be counteracted with the bias since it operates independently of the inputs. It is therefore not possible for our printed neural network to map both classes arbitrarily far apart in the last layer, i.e. to cover the range between 0 and 1 (or -1 and 1). We therefore resort to a special variant of the Hinge-Loss (also called SVM-loss or maximum-margin-loss):

$$L_{Hinge}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \max\{0,\ m - y \cdot f_\theta(x)\},$$

where $m \in \mathbb{R}^+$ denotes the so called margin and $y \in \{-1, 1\}$ denotes the true label. A positive loss is incurred, when the output of the neural network is either of the wrong label, i.e. $y \neq sign(f_\theta(x))$, or the output $f_\theta(x)$ is smaller than the margin m. For correct outputs, i.e. $y = sign(f_\theta(x))$, and $f_\theta(x) > m$, no error is incurred. Using $m = 1$, the classical Hinge-Loss is realized. As discussed before, the range of our outputs is ever shrinking through successive layers and it is not possible to set a fixed value of $m > 0$, as we cannot guarantee that outputs of different classes can be mapped further apart than m. We therefore choose $m = 0$, which will result in our loss function only penalizing wrongly classified examples, while correctly classified examples produce no loss.

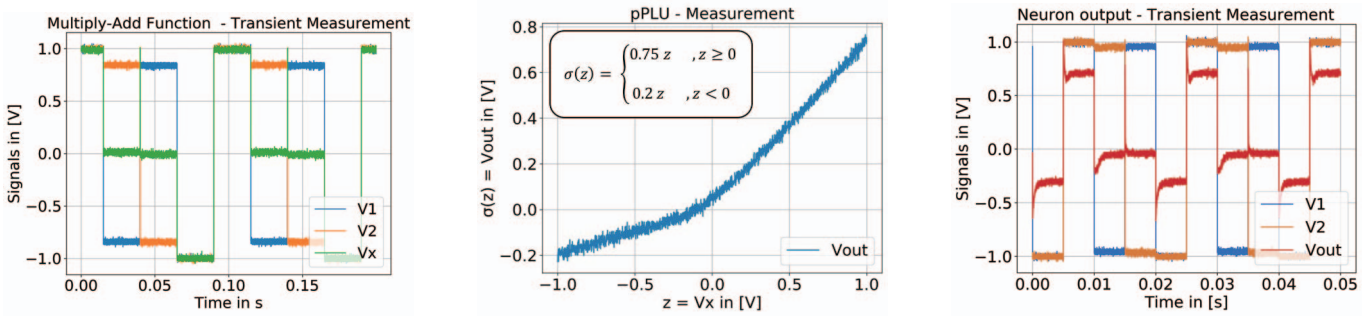
Furthermore, to slightly reduce the incurring voltage drop due to small weights, one can add the negative norm of the weight vectors $w^{(n)}$ to the loss to encourage higher weights. This leads to a final loss function of

$$L(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \max\{0,\ -y \cdot f_\theta(x)\} - \lambda \sum_{\forall n} \|w^{(n)}\|_2,$$

where $\lambda \in \mathbb{R}^+$ denotes some tunable parameter. Note that due to the boundedness of the weights, the loss is still bounded.

B. Back Propagation with Projected Gradients

Since neural networks are mostly trained using gradient-based methods and back-propagation, we will adapt the update-steps accordingly to satisfy the given constraints via projection on the feasible set. In the following, we denote



(a) (b) (c)

Fig. 4: Measurement results of inkjet printed 2-input neuron, a): multiply-add-function for $w_1 = w_2 = 0.5$ b): measured activation function c): neuron output voltage V_{out} with suppressed response for negative input voltage combination ($V_1 = V_2 = -1V$)

parameters of a neuron n by $\theta^{(n)} = [w^{(n)},\ b^{(n)}]^T \in \mathbb{R}^{n_P+1}$ for ease of notation. An update step using the step-length α can then be expressed as:

$$\theta'^{(n)} \leftarrow \theta^{(n)} - \alpha \cdot \nabla_{\theta^{(n)}} L(\theta^{(n)})$$
$$\theta^{(n)} \leftarrow \mathit{Proj}\left[\theta'^{(n)}\right],$$

where the projection function $\mathit{Proj}\left[\theta'^{(n)}\right]$ returns the closest feasible solution vector to $\theta'^{(n)}$ in euclidean norm. The solution of the projection problem can be obtained by solving the following optimization problem:

$$\mathit{Proj}\left[\theta'^{(n)}\right] := \underset{[w,\,b]^T}{\operatorname{argmin}} \left\| \theta'^{(n)} - \begin{bmatrix} w \\ b \end{bmatrix} \right\|_2 \quad \text{s.t.}$$
$$\sum_{i=1}^{n_P} w_i \leq 1 - w_b^{min}$$
$$0 \leq w_i \leq 1 - w_b^{min}, \quad \forall i = 1, \cdots, n_P$$
$$-1 \leq b \leq 1$$

As this projection is a convex optimization problem due to a norm being minimized over a non-empty, closed convex set [13], it is always solvable and the solution is unique. Through projection, the parameters after each update step will always satisfy the given constraints.

In our case, the projection problem is solved using Sequential Quadratic Programming [14] implemented in *scipy* [15], and the gradients of the neural network are retrieved using *pytorch* [16].

C. Evaluation on 2-layer Neural Network

To verify the feasibility of the proposed design, we train a neural network on a simple classification task given the aforementioned characteristics i.e. parameter constraints, activation function and loss function. The neural network is then converted into its circuit representation and the classification errors are evaluated based on a circuit simulation in Cadence Virtuoso.

For the classification task, we use the IRIS flower data set [17, 18] containing 150 four dimensional examples of three classes (50 per class), and try to distinguish the first class (*setosa*) versus the other two (*versicolor* and *virginica*). Note that multi-class classification is not yet possible as we cannot fabricate a softmax output. We therefore perform 1-vs-rest classification choosing class 1 (*setosa*) if V_{OUT} is positive and class 2 (*versicolor* and *virginica*) otherwise. As the dataset has four inputs, we choose the architecture of the neural network as follows: two five-inputs nodes, i.e. four weights and a bias for printed neuron node in the first layer, and one 3-input multiply-add function, i.e. two weights and one bias, in the last layer (Fig. 5). Since the task is linearly separable, a close to 100 % accuracy can be achieved in software-based training.

TABLE II: Simulation Results - Neural Network Classification

Max Voltage	Accuracy	Latency	Max Operating Frequency	Power	Area
5V	88%	1.6ms	625 Hz	4.3mW	385 mm^2

For the circuit simulation, the input vector $\vec{x}$ is converted into input voltages: $\vec{x} = (V_1, V_2, ..., V_4)^T$. Subsequently, the crossbar resistance values $R_i^{(n)}$ are directly computed from the optimal weights $w_i^{(n)}$ obtained from the training routine. We used a symbolic solver (*sympy*) on the set of coupled equations of the form (3) in order to receive the $R_i^{(n)}$. $V_{bias}^{(n)}$ are computed by $V_{bias}^{(n)} = \frac{b^{(n)}}{w_b^{(n)}}$.

Simulation results are illustrated in Table II. Due to approximations of the printed NCS components (multiply-add, pPLU) from the exact analytical function descriptions used during in-software training - which usually persist in any analog computing system - circuit simulation accuracy was lowered. Nevertheless, still a classification accuracy of 88% could be achieved. The latency and thus the maximum operating frequency, which are mainly caused by the EGFET gate capacitance in the pPLU, was equal to the fabricated 2-input neuron (Section IV), as only one activation function layer was deployed in the NN.

VI. Conclusion and Future Work

In this work, we reported on a programmable neuron for neuromorphic computing systems (NCS). The neuron can fully be tuned during post-fabrication customization according to the desired functionality. As mapping of NCS computations to physical hardware is accompanied by many design constraints, especially the limitation to positive weights, low device count

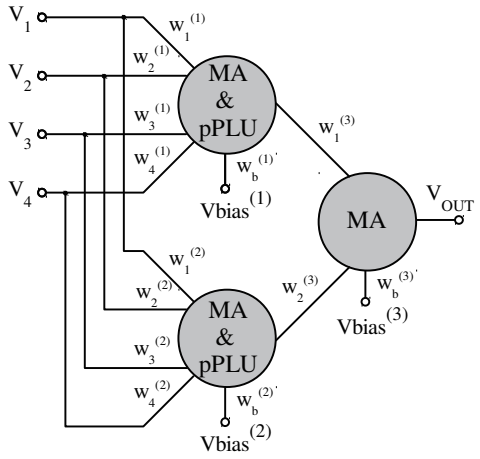


Fig. 5: Architecture of the neural network: two 5-input neurons with pPLU-activation function in the first layer and one 3-input neuron with only multiply-add operation (MA) in the second layer.

and NMOS-logic in this emerging technology, we developed a learning algorithm for inkjet-printed programmable NCS, which takes all these technology and hardware restrictions into account to produce a feasible and well-performing realization. Finally, the learning routine was verified on a simulation-based printable neural network design solving a common classification problem.

Although the objective of this paper is to prove the feasibility of the printed neuron concept, we still see a lot of room for improvements. ITO-sputtered passive conductive tracks were the preferred choice to build a benchmark circuit showing the potential of the printed neuron and secondly, used to reduce process variations. These conductive elements can be replaced in the future by printed conductive materials such as PEDOT:PSS to enable a fully additive process. Furthermore, with a fully inkjet-printed design, process variations of resistors can be compensated by introducing a second printing step after the first fabrication and characterization. And finally, another improvement of the current approach would be the inclusion of printable negative weights to broaden the scope of neural network applications.

Acknowledgements

This work was supported by the Ministry of Science, Research and Arts of the state of Baden-Württemberg in form of the MERAGEM doctoral program.

References

[1] Michael Schmuker, Thomas Pfeil, and Martin Paul Nawrot. “A neuromorphic network for generic multivariate data classification”. In: *Proceedings of the National Academy of Sciences* 111.6 (2014), pp. 2081–2086.

[2] Paul A Merolla et al. “A million spiking-neuron integrated circuit with a scalable communication network and interface”. In: *Science* 345.6197 (2014), pp. 668–673.

[3] Joseph S Chang, Antonio F Facchetti, and Robert Reuss. “A circuits and systems perspective of organic/printed electronics: review, challenges, and contemporary and emerging design approaches”. In: *IEEE Journal on emerging and selected topics in circuits and systems* 7.1 (2017), pp. 7–26.

[4] Gabriel Cadilha Marques et al. “Progress Report on ”From Printed Electrolyte-Gated Metal-Oxide Devices to Circuits””. In: *Advanced Materials* (2019), p. 1806483.

[5] Dennis Weller et al. “An inkjet-printed low-voltage latch based on inorganic electrolyte-gated transistors”. In: *IEEE Electron Device Letters* 39.6 (2018), pp. 831–834.

[6] Ahmet Turan Erozan et al. “Inkjet-printed EGFET-based physical unclonable functionDesign, evaluation, and fabrication”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 26.12 (2018), pp. 2935–2946.

[7] Miao Hu et al. “Hardware realization of BSB recall function using memristor crossbar arrays”. In: *Proceedings of the 49th Annual Design Automation Conference*. ACM. 2012, pp. 498–503.

[8] Wei Wen et al. “An EDA framework for large scale hybrid neuromorphic computing systems”. In: *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE. 2015, pp. 1–6.

[9] Yandan Wang et al. “Classification accuracy improvement for neuromorphic computing systems with one-level precision synapses”. In: *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE. 2017, pp. 776–781.

[10] Yoeri van De Burgt et al. “Organic electronics for neuromorphic computing”. In: *Nature Electronics* (2018), p. 1.

[11] Robert A Nawrocki, Richard M Voyles, and Sean E Shaheen. “Neurons in polymer: Hardware neural units based on polymer memristive devices and polymer transistors”. In: *IEEE Transactions on Electron Devices* 61.10 (2014), pp. 3513–3519.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[13] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[14] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. 2nd ed. Springer Science & Business Media, 2006.

[15] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. accessed 22.06.2019. 2001–. URL: http://www.scipy.org/.

[16] Adam Paszke et al. “Automatic differentiation in PyTorch”. In: *NIPS-W*. 2017.

[17] Edgar Anderson. “The species problem in Iris”. In: *Annals of the Missouri Botanical Garden* 23.3 (1936), pp. 457–509.

[18] Ronald A Fisher. “The use of multiple measurements in taxonomic problems”. In: *Annals of eugenics* 7.2 (1936), pp. 179–188.