Reminding Forgetful Organic Neuromorphic Device Networks

Daniel Felder
DWI - Leibniz Institute for Interactive Materials, Forckenbeckstrasse 50, Aachen, DE
AVT.CVT - Chair of Chemical Process Engineering, RWTH Aachen University,
Forckenbeckstrasse 51, Aachen, DE

Katerina Muche
AVT.CVT - Chair of Chemical Process Engineering, RWTH Aachen University,
Forckenbeckstrasse 51, Aachen, DE

John Linkhorst
AVT.CVT - Chair of Chemical Process Engineering, RWTH Aachen University,
Forckenbeckstrasse 51, Aachen, DE

Matthias Wessling
DWI - Leibniz Institute for Interactive Materials, Forckenbeckstrasse 50, Aachen, DE
AVT.CVT - Chair of Chemical Process Engineering, RWTH Aachen University,
Forckenbeckstrasse 51, Aachen, DE
E-mail: manuscripts.cvt@avt.rwth-aachen.de

Abstract. Organic neuromorphic device networks can accelerate neural network algorithms and directly integrate with microfluidic systems or living tissues. Proposed devices based on the bio-compatible conductive polymer PEDOT:PSS have shown high switching speeds and low energy demand. However, as electrochemical systems, they are prone to self-discharge through parasitic electrochemical reactions. Therefore, the network’s synapses forget their trained conductance states over time. This work integrates single-device high-resolution charge transport models to simulate entire neuromorphic device networks and analyze the impact of self-discharge on network performance. Simulation of a single-layer nine-pixel image classification network commonly used in experimental demonstrations reveals no significant impact of self-discharge on training efficiency. And, even though the network’s weights drift significantly during self-discharge, its predictions remain 100% accurate for over ten hours. On the other hand, a multi-layer network for the approximation of the circle function is shown to degrade significantly over twenty minutes with a final mean-squared-error loss of 0.4. We propose to counter the effect by periodically reminding the network based on a map between a synapse’s current state, the time since the last reminder, and the weight drift. We show that this method with a map obtained through validated simulations can reduce the effective loss to below 0.1 even with worst-case assumptions. Finally, while the training of this network is affected by self-discharge, a good classification is still obtained. Electrochemical organic neuromorphic devices
have not been integrated into larger device networks. This work predicts their behavior under nonideal conditions, mitigates the worst-case effects of parasitic self-discharge, and opens the path toward implementing fast and efficient neural networks on organic neuromorphic hardware.

Keywords: neuromorphic computing, artificial synapse, organic electronics, neural network, algorithm-hardware co-design

1. Introduction

Neuromorphic devices can accelerate large neural networks to unlock new applications in natural language processing [1], image generation [2] and many more. At the same time, the energy efficiency of neuromorphic devices allows the integration of small neural networks into low-powered devices for edge applications [3]. Organic neuromorphic devices allow, furthermore, a direct integration into electrolytic systems and can directly communicate with interactive materials such as artificial cells through ion channels, microgels through ion release and sensing [4, 5], and living tissues such as nerve cells [6]. These properties open a pathway towards intelligent systems that combine sensing, processing, and memory [7, 8].

Recently several organic neuromorphic device concepts have been presented. Van de Burgt et al. demonstrated electrochemical random access memories (ECRAMs) [9] that have been adapted to work at elevated temperatures [10] or to be screen printable [11]. Lee et al. propose a similar device with a nanofiber-based channel and ultra-fast response times [12].

ECRAMs based on PEDOT:PSS and a solid-state electrolyte are among the very promising candidates. They feature high write linearity for efficient neural network training algorithms, low energy demand, and decoupled read and write operations through a separate gate electrode. However, they have been shown to exhibit self-discharge, which makes a neural network’s weights gradually drift toward zero over time. While physical mitigation strategies have been proposed, perfect self-discharge-free devices are difficult to produce, especially for scaled-down versions and devices that are supposed to interact with other interactive material systems. This weight drift with long timescales can be useful in some scenarios like eligibility traces in reinforcement learning [13]. For conventional, pre-trained neural networks, it results, however, in a gradual degradation in prediction performance that needs to be precisely understood.

This work uses comprehensive numeric models to understand the effect of self-discharge on neural network performance and enable its mitigation through efficient algorithm-hardware co-design. It is shown that some networks are highly vulnerable to self-discharge, while others are completely immune. Furthermore, algorithmic countermeasures are proposed to stabilize vulnerable networks for extended amounts of time.
2. Methods

This work aggregates physically accurate single-device models into multiple device arrays. Each array represents a neural network layer and is simulated transiently for backpropagation and inference.

2.1. Single-device model

The single-device model is based on Nernst-Planck-Poisson two-phase charge transport modeling of an electronically conductive PEDOT phase and an ionically conductive PSS and electrolyte phase. Both are coupled capacitively with an approach first presented by Tybrandt et al. [15]. Additionally, self-discharge is described through an electrochemical shuttle reaction modeled with a Butler-Volmer kinetic, and water dissociation is described by mass action law kinetics. Figure 1A visualizes the single device with the modeled effects. Writing and reading are decoupled during neuromorphic operations. Therefore, lateral ion transport in the channel can be neglected, and the modeling domain reduced to a 1D line between the gate and drain electrode. These assumptions also hold when devices are integrated into crossbars and are valid as long as low read potentials and sufficient relaxation times between reads and writes are used. The single-device model’s equations, characteristics, and validation were described in detail in a previous publication [14].

Figure 1B visualizes single-device programming. Increasing potentials at the gate electrode result in lower conductances in the channel, while decreasing potentials
increase the conductance. The devices can be switched reproducibly between many discrete states \cite{9}. However, in the presence of impurities such as ambient oxygen, these states quickly decay towards the initial device conductance. The effect is caused by an electrochemical shuttle reaction between the two PEDOT:PSS layers. One of the identified reactions is a hydrogen peroxide shuttle \cite{16}. As the potential increases, the reaction and, therefore, the state decay gets exponentially faster. This results in highly stable states close to the initial conductance and much less stable states for larger or smaller conductances.

Figure 1C shows a comparison of real-world experimental data of self-discharge with the applied model. States with an initial gate potential around $\pm 0.25$ V decay insignificantly over 10 min, while states with an initial potential around $\pm 0.5$ V decay by up to 12\% of the total range in the same time. The data highlight an additional nonideality that has to be considered for some devices. Nonlinear conductance tuning causes the states in the figure to lie closer together for negative potentials. This is caused by crowding in the density of states that results in a lowered electron mobility and is accounted for in the model through an experimentally obtained correlation. Mitigation strategies through partial imine dededoping have been successfully employed in literature \cite{9, 17, 18}. Therefore, ideal linear devices are assumed unless otherwise noted in the following.

2.2. Crossbar model

To manufacture large scale device networks, single devices are commonly integrated into crossbar structures. For two-terminal devices, a set of parallel leads connects to the devices’ bottom terminal with another set of leads perpendicular to the first connecting the top electrode. This results in a configuration where every device can be addressed individually for programming and up to a full row can be programmed at the same time. For electrochemical neuromorphic devices with three terminals, such an architecture can be adapted to include a third set of leads. Figure 2B visualizes this concept with a crossbar built with differential artificial synapses. Differential synapses compensate for fluctuations in the devices’ characteristics during fabrication by combining two devices and subtracting the devices’ resulting currents from each other. During inference, each device in the differential synapse receives the same voltage inputs. During training, the devices are programmed into opposite directions with the same but inverted gate voltages. To map device conductance to neural network weights, they are normalized with a reference initial conductance $G^{0,\text{ref}}$. Each device’s gate voltage range is limited to between $-0.5$ V and $0.5$ V. Therefore, weights are capped at -1 and +1. The neural network algorithms can partially compensate for this limitation through a weight-scale factor $\beta \geq 1$. Networks with bounded weights have previously been studied to limit overfitting at the cost of reduced learning capability \cite{19}. Therefore, differential weights
$W_{ij}$ with the following formula are created.

$$W_{ij} = \beta \cdot \frac{G^+_{ij} - G^-_{ij}}{G_{0,\text{ref}}}$$  \hspace{1cm} (1)

where $G^\pm_{ij}$ is the devices’ conductance.

When single-device models are placed in a crossbar configuration, their potential interactions must be considered. The devices themselves do not directly influence each other. However, the length of the leads to the power source and transimpedance amplifier changes from device to device resulting in variable parasitic resistances. It has been shown that the channel conductance of the devices can be finely tuned through partial imine dedoping of PEDOT [20]. We, therefore, assume that it is possible to build devices with significantly higher channel resistance than the leads and expect the lead resistances to be negligible during read operations. For writing, the gate to channel impedance cannot be easily modified, but programming pulses can be chosen long enough to get close to steady-state independently of parasitic resistances. Therefore, the crossbar is modeled as a network of single-device models with ideal connections.

2.3. Inference

During a forward pass through each network layer, inputs are encoded as continuous voltages $V_j$ and scaled to the range of $\pm 0.1$ V by multiplication with $\gamma = 0.1$ V Limiting of input voltages of consecutive layers is achieved through the corresponding activation functions. With Kirchhoff’s law the following output current results for each neuron in each layer

$$I_\pm^i = \sum_{j=1}^{N} V_j G^\pm_{ij}$$  \hspace{1cm} (2)

where $N$ is the number of neurons in the previous layer. The currents are converted back into voltages by a transimpedance amplifier with a rate of $\beta \cdot (G_{0,\text{ref}}^{-1})$, subtracted and passed through a hardware implementation of the $f(x) = \gamma \cdot \tanh(\gamma^{-1} \cdot x)$ activation function. An ideal implementation of the activation function is assumed to focus on the unique properties of organic neuromorphic devices as synapses. $\tanh$ is chosen to maintain comparability with Prezioso’s work [21], and because it is symmetric and zero-centered. The minimal current range of the transimpedance amplifier is given by $I_{\text{max}} = N \cdot 0.1 \text{V} \cdot G|_{V_\text{g}=-0.5 \text{V}}$. The voltages at the output layer are finally scaled back up by a factor of $\gamma^{-1}$ to obtain real-valued outputs.

2.4. On-device training

On-device training allows the neural networks to adapt to manufacturing variability between the different neuromorphic devices and, therefore, partially compensate it. Furthermore, on-device training benefits from the same highly parallel multiply and accumulate operations as the inference step. Here, a modified, discrete Delta Rule
Figure 2. Neural network hardware implementation. (A) Crossbar of organic neuromorphic devices with differential synapses. Input potentials are applied on the left, gate potentials on the top, and resulting currents are read on the bottom. (B) Example network for the crossbar in A that classifies 3x3 pixel images into the categories z, v, and n. (C) Input data to the example network. Ideal pictures on the left, noisy pictures on the right.

approach known as Manhattan Rule is applied following Prezioso et al.’s demonstration of training on metal oxide devices [21, 22].

Weights are initialized by random, normally distributed 1 s pulses between $-0.5 \text{ V}$ and $0.5 \text{ V}$. 1 s pulses were previously shown to be long enough for capacitive charging processes to reach a near-equilibrium state and short enough to suppress electrochemical reactions. If weight-scaling is applied through $\beta > 1$, the pulses are divided by $\beta$ before being constrained to the maximum gate potential of $\pm 0.5 \text{ V}$. Weight updates are performed through 100 ms pulses with positive or negative potentials versus the device’s current open circuit potential (OCP). In Manhattan Rule training, each update pulse has the same fixed height $\eta$ equivalent to a learning rate. Mean-squared-error (MSE) is used as loss function. For the output layer, weight increments $\Delta_{ij}(n)$ for each data point $n$ are calculated as

$$\delta_i(n) = [t_i(n) - y_i(n)] \cdot \tanh'(a_i(n)) \quad (3)$$
$$\Delta_{ij}(n) = \delta_i(n)x_j(n) \quad (4)$$

where $x$ are the input data, $t$ the target values, $a$ the layer’s outputs before the activation function, and $y$ after the activation function. $\Delta_{ij}(n)$ are computed for one $n$ at a time and aggregated. After all inputs have been applied, the synaptic weights are updated by the Manhattan Rule.

$$\Delta W_{ij} = \eta \cdot \text{sgn} \sum_{n=1}^{N} \Delta_{ij}(n) \quad (5)$$

For multi-layer networks, the backpropagation algorithm is applied for the
where \( l \) denotes the layer number from input towards output layer. Here, the backpropagation step \( \delta_{l+1} \cdot \beta W_l^\top \) is applied through reversal of the crossbar. Crossbar reversal is described in detail in the Supporting Information section S1.

### 2.5. Self-discharge simulation

After virtual on-device training, a model for every neuromorphic device in every layer is obtained with an individual conductance state and concentration distribution. During the self-discharge simulation, these device models are advanced through time in open circuit mode. No electrical current flows between the source or drain and the gate electrode. Changes result purely from the electrochemical shuttle reaction and the resulting ion movement. Channel conductances are calculated from the channel’s electronic charge carrier concentration and monitored over time. Every 20 s the networks’ loss functions are evaluated to monitor prediction performance decline.

### 2.6. Device reminders

In contrast to, for example, PCM devices [23], the conductance drift of organic devices is a complex function of the device’s state and, therefore, more difficult to compensate. However, with the presented model, each device’s conductance decline can be precisely predicted depending on its history and initial state. These predictions are used to design reminder pulses to reverse the effects of self-discharge. For an efficient implementation, a calibration map between the initial synapse weight, the time of self-discharge and the current weight is created. Data is produced by simulating synapses for a range of starting weights \( w_{\text{start}} \in [0, 0.1, 0.2, ..., 0.9, 1] \) for 20 000 s. Since differential synapses are symmetric, no negative starting weights are required. The data is then transformed into a map between the current weight, the time of self-discharge and the resulting delta in the weight’s value. This map is interpolated in 2D to obtain weight deltas for arbitrary points within the data limits.

### 2.7. Read-error effect estimation

To estimate the effect of read-errors on inference and backpropagation, weight sets with normally distributed read errors are created during each read operation. It is assumed that the transimpedance amplifiers do not have a systematic error and the mean of the distribution is zero, the standard deviation is based on Keene’s observations [17]. The normal distribution is chosen as default lacking more experimental data.

\[
W_{ij,\text{read}} = W_{ij,\text{ideal}} + \text{err}_{ij} \tag{8}
\]

\[
\text{err}_{ij} \sim \mathcal{N}(0, 0.0025^2) \tag{9}
\]
During backpropagation, the weights are sampled with new errors for each inference and backpropagation step.

3. Results

The effect of forgetful neuromorphic devices is analyzed for two example neural networks. One is a single-layer perceptron designed to recognize 3x3 pixel images as letters z, n, and v. Similar networks are often used for the demonstration of real-world neuromorphic devices [24,21]. The other is a multi-layer network to approximate the circle function based on points in a 2D space. For both, good prediction performance is obtained with simulated on-device training. While the single-layer perceptron seems completely immune to the effects of self-discharge, the multi-layer network is affected strongly.
3.1. Single-layer image classification

The single-layer image classification network schematically drawn in Figure 2A is designed to recognize 3x3 pixel representations of the letters z, v, and n. The nine pixels are encoded to -1 for black and 1 for white. As an additional 10th input -1 is passed into the device network to replace the output neurons’ biases by the weights between the 10th input and the corresponding output neuron without further adjustment to the crossbar structure. The network is trained on the images drawn in Figure 2C. Additionally to the perfect representations on the left, noise is introduced by flipping one pixel in each additional image [21]. In view of the small size of the dataset, no distinction between training and test set is made.

Figure 3A shows multiple simulated on-device training cycles with self-discharge simulated at ambient oxygen concentrations. Each cycle starts from a different set of normally-distributed random weights but is otherwise identical. Consistently, less than ten epochs are required to reach perfect classification of the training set. This is comparable to training cycles on metal oxide devices presented by Prezioso et al. where between 5 and 35 epochs are required [21]. The resulting weights (Figure 3B) are, for a randomly picked run, relatively equally distributed with synapses connecting to n having slightly larger values on average. Drift for the first 10h is strongest for the large weights around ±0.6 and reaches up to 30% of the total range. For smaller weights up to ±0.1, drift is inconsequential with 1% of the total range. As Figure 3C, however, shows, even after 10h and significant weight drift, no classification errors are introduced. At the same time, the network’s loss rises from 0.14 to 0.24.

We hypothesize that this highly stable network behavior is caused by two factors. First, the network only has a single layer and nonlinearity is only introduced at the output. Every weight drifts in the same direction towards zero, but the relative order of weight sizes is preserved. Therefore, no single output neuron is favored, and the predictions for each image are unlikely to change. Second, the network has many active synapses connected to each output neuron which results in large input values to the activation function of each output neuron. Even after a long self-discharge duration, the low gradient in the tanh activation function for large values allows only small increases in the mean-squared-error loss.

3.2. Multi-layer regression network

As second example, a multi-layer network for the approximation of the circle function \( r^2 = x_1^2 + x_2^2 \) is analyzed (similar to [25]). The network includes nonlinear effects through tanh activation function and two hidden layers. It has an even shape with two input neurons, four and two neurons in the hidden layers and one output neuron as drawn in Figure 4B. Thus, a stronger reaction to self-discharge is expected compared to the single-layer perceptron.

Data points are randomly generated in 2D \((x_1, x_2)\) and classified by whether they lie inside or outside a circle with the radius 0.5, see Figure 4A. No data points
with a radius between 0.5 and 0.7 are generated to allow a clearer separation of the two classes. Initially, training without self-discharge with the Manhattan Rule is attempted and results in good classification rates after 60 to 150 epochs as shown in Figure 4C. Here, a step function learning rate scheduler is applied to increase convergence speed.

$$\eta = 0.03 \cdot 0.7 \left[ \frac{1 + \text{epoch}}{20} \right] \cdot \beta^{-1}$$

During inference, a drastic effect of self-discharge is shown over a time frame of 20 min. For ambient oxygen concentrations, an increase in the network’s loss function from 0.01 to 0.41 is observed and drawn in Figure 4D. For a 90% reduced oxygen concentrations, the loss still rises to 0.08. Figure 4F visualizes the change in the...
network’s predictions by drawing the decision boundary as a red and blue background behind the training data points in red (inside the circle) and blue (outside the circle) in the foreground. After 10 min a large portion of the outside points is misclassified as inside the circle.

3.3. Reminding forgetful device networks

As the previous section shows, drift in forgetful neuromorphic devices can quickly degrade the prediction performance of neural networks. With the single device charge transport model, this drift can be precisely predicted and reversed through engineered reminder pulses (see methods section). Figure 4E shows in the inset the reset map between the current synapse weight, the time of self-discharge, and the delta pulse that is required to reset each synapse to its original state. A quantitative version of the inset can be found in the supporting information Figure S2. Since differential synapses are used, each synapse’s behavior is perfectly symmetric around zero, even if a single device’s behavior is not. Therefore, only the positive weight range is calculated and shown in the figure. For weights close to zero or after very short self-discharge times, almost no corrections are necessary. For times of 20 000 s and weights approaching 1, the delta approaches 0.63. The main graph shows how the engineered reminder pulses affect the network’s MSE loss function over time. A device network trained to approximate the circle function that self-discharged for 20 min is taken as base state. At 0 s the first reminder is applied to the network and results in a decrease in loss from 0.41 to 0.01. Applying the same procedure every 100 s continuously keeps the network’s loss below 0.06 even with worst-case assumptions about device stability. While only virtual device reminders were tested in this work, the used models have been shown to validate well against real-world devices. Therefore, the presented approach is expected to transfer to real-world devices. The additional energy consumption by the reminders can be expected to be well below 10% of the ideal network’s consumption. A detailed estimation is provided in the supporting information, section S5.

Training forgetful device networks

Given the vulnerability of the multi-layer network towards self-discharge during inference, the influence of self-discharge and nonlinearity during training is evaluated separately. Figure 5A shows the loss function during training with and without self-discharge. Here, the same initialization is chosen for the weights to allow a one-to-one comparison. During the initial epochs up until epoch 25, only small deviations in the training progress can be observed. After epoch 25, the effect of self-discharge becomes visible, with the ideal approaching 0.002 until epoch 60 in the reference case. With self-discharge, a limit of 0.014 is approached after 60 epochs for the same initial weights. Figure 5B shows the network’s weight distribution for epoch 20, 40, and 60 during nonideal training. Especially in the output layer (layer 3), weight values are pushed towards large values. These large values cause a large vulnerability towards self-
discharge also during training. Adapting the weight scaling for larger possible values by doubling $\beta$ to 4 allows the device conductance values to remain closer to the initial state and, therefore, remain more stable. Here, a trade-off has to be made. Using a smaller part of the conductance range reduces the effects of self-discharge. At the same time, the effects of read and write errors start to become more dominant. In online training, write errors are eventually canceled out. Read errors, however, can a) affect the final accuracy of the network and b) affect backpropagation performance. Figure 5C shows simulations with a commonly assumed [17] read error of 0.5% for $\beta = 2$ and $\beta = 4$. The increased stochasticity affects increase initial convergence but also limits the final accuracy for $\beta = 4$. Here, the accuracy gains from reduced self-discharge are completely neutralized. Therefore, training can, in this case, not be enhanced through $\beta$-scaling. Instead, more synapses and a modified network structure would be required to decrease the loss further. However, the resulting final loss still leads to good classification in this case.

Training with nonlinear devices, on the other hand, results in large residual errors during training, see Supporting Information Figure S3. Even with differential devices, update symmetry is not perfect, and the total conductance range available is reduced. While asymmetry can be compensated algorithmically [26], state-of-the-art devices achieve highly linear conductance switching through partial imine dedoping and should be preferred [20].

4. Discussion

Many neuromorphic devices are vulnerable to self-discharge and weight drift. At the same time, conventional neural network algorithms are often not well-suited to
compensate these effects. We have demonstrated a model-based approach to quickly assess these effects for organic, electrochemical device networks and develop mitigation strategies.

The approach was applied to two example networks. The single-layer perceptron for 3x3 image classification showed good stability both during training as well as inference for extended self-discharge times of up to 10 h. We hypothesize that a combination of a large number of active synapses per output neuron and no effective nonlinearity in the network leads to the network’s resilience against weight drift.

Training a three-layer network on the circle function reveals a much stronger influence of weight drift. Training of the network requires less than 3 min for 60 epochs with 100 ms update pulses. Therefore, only small weight changes through self-discharge are expected. At the same time, on-device training is able to react to and correct the weight’s drift. Still, a reduced convergence speed and a larger final loss of 0.014 instead of 0.002 without self-discharge are observed. There exists potential to optimize the weight distribution towards smaller conductance ranges and, therefore, reduce self-discharge that affects large weights most strongly. However, a trade-off with the hardware’s read-accuracy has to be considered. During inference, weight drift has a significant effect and degrades the network’s prediction performance to a loss of 0.41 after 20 min for devices with uninhibited self-discharge. We propose a method of stabilizing these networks by periodic reminder pulses. These pulses can be effectively designed with the device model and require only the current device state and the time since the last reminder pulse as input to correct device drift. In the example network, pulses once per 100 s are shown to consistently keep the loss below 0.06 without impacting the network’s speed or energy efficiency. This is an important result, as it shows that the limitations of organic neuromorphic hardware can be efficiently mitigated through algorithm-hardware co-design.

Algorithm-hardware co-design can unlock further improvements through network optimization. E.g., as the number of weights in a network increases, their magnitude can be decreased \[24, 27, 28\]; in the extreme, even binarized networks are possible \[29\]. Additionally, effective always-on, unsupervised learning methods can eliminate the need for artificial device reminders \[30, 31\] and even turn self-discharge into a useful platform feature to take full advantage of the devices’ properties.

We conclude that neural networks on forgetful neuromorphic hardware can be effectively implemented but require adjusted algorithms whose development is aided by model-based design.

5. Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.
6. Acknowledgements

Matthias Wessling acknowledges DFG funding through the Gottfried Wilhelm Leibniz Award 2019 (WE 4678/12-1).

References

[1] Romal Thoppilan et al. “Lamda: Language models for dialog applications”. In: arXiv preprint arXiv:2201.08239 (2022).
[2] Chitwan Saharia et al. “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”. In: arXiv preprint arXiv:2205.11487 (2022).
[3] Dennis Valbjørn Christensen et al. “2022 roadmap on neuromorphic computing and engineering”. In: Neuromorphic Computing and Engineering (2022).
[4] Sanggil Han et al. “Microfabricated Ion-Selective Transistors with Fast and Super-Nernstian Response”. In: Advanced Materials 32.48 (2020), p. 2004790.
[5] Shao-Tuan Chen, Christopher M Proctor, and George G Malliaras. “Materials and Device Considerations in Electrophoretic Drug Delivery Devices”. In: Scientific reports 10.1 (2020), pp. 1–7.
[6] Scott T Keene et al. “A biohybrid synapse with neurotransmitter-mediated plasticity”. In: Nature Materials 19.9 (2020), pp. 969–973.
[7] Yoeri van de Burgt. Brain-Inspired Organic Modular Lab-on-a-Chip for Cell Classification. https://dx.doi.org/10.3030/802615 2019.
[8] C Kaspar et al. “The rise of intelligent matter”. In: Nature 594.7863 (2021), pp. 345–355.
[9] Yoeri van de Burgt et al. “A non-volatile organic electrochemical device as a low-voltage artificial synapse for neuromorphic computing”. In: Nature materials 16.4 (2017), pp. 414–418.
[10] Armantas Melianas et al. “Temperature-resilient solid-state organic artificial synapses for neuromorphic computing”. In: Science advances 6.27 (2020), eabb2958.
[11] Qingzhou Liu et al. “Fully printed all-solid-state organic flexible artificial synapse for neuromorphic computing”. In: ACS applied materials & interfaces 11.18 (2019), pp. 16749–16757.
[12] Sol-Kyu Lee et al. “Nanofiber Channel Organic Electrochemical Transistors for Low-Power Neuromorphic Computing and Wide-Bandwidth Sensing Platforms”. In: Advanced Science (2021), p. 2001544.
[13] Yi˘git Demira˘g et al. “PCM-Trace: scalable synaptic eligibility traces with resistivity drift of phase-change materials”. In: 2021 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE. 2021, pp. 1–5.
Daniel Felder et al. “Coupled Ionic-Electronic Charge Transport in Organic Neuromorphic Devices”. In: Advanced Theory and Simulations (2022).

Klas Tybrandt, Igor V Zozoulenko, and Magnus Berggren. “Chemical potential–electric double layer coupling in conjugated polymer–polyelectrolyte blends”. In: Science advances 3.12 (2017), eaao3659.

Musbaudeen O Bangbopa et al. “Modelling of heterogeneous ion transport in conducting polymer supercapacitors”. In: Journal of Materials Chemistry A 9.4 (2021), pp. 2184–2194.

Scott Tom Keene et al. “Mechanisms for Enhanced State Retention and Stability in Redox-Gated Organic Neuromorphic Devices”. In: Advanced Electronic Materials 5.2 (2019), p. 1800686.

Scott T Keene et al. “Enhancement-Mode PEDOT: PSS Organic Electrochemical Transistors Using Molecular De-Doping”. In: Advanced Materials 32.19 (2020), p. 2000270.

Yi Liao, Shu-Cherng Fang, and Henry LW Nuttle. “A neural network model with bounded-weights for pattern classification”. In: Computers & Operations Research 31.9 (2004), pp. 1411–1426.

Tom PA van der Pol et al. “The mechanism of dedoping PEDOT: PSS by aliphatic polyamines”. In: The Journal of Physical Chemistry C 123.39 (2019), pp. 24328–24337.

Mirko Prezioso et al. “Training and operation of an integrated neuromorphic network based on metal-oxide memristors”. In: Nature 521.7550 (2015), pp. 61–64.

Elham Zamanidoost et al. “Manhattan rule training for memristive crossbar circuit pattern classifiers”. In: 2015 IEEE 9th International Symposium on Intelligent Signal Processing (WISP) Proceedings. IEEE. 2015, pp. 1–6.

Stefano Ambrogio et al. “Reducing the impact of phase-change memory conductance drift on the inference of large-scale hardware neural networks”. In: 2019 IEEE International Electron Devices Meeting (IEDM). IEEE. 2019, pp. 6–1.

I Giannopoulos et al. “8-bit precision in-memory multiplication with projected phase-change memory”. In: 2018 IEEE International Electron Devices Meeting (IEDM). IEEE. 2018, pp. 27–7.

Daniel Smilkov and Shan Carter. Tensorflow Playground. https://github.com/tensorflow/playground 2022.

Tayfun Gokmen and Wilfried Haensch. “Algorithm for training neural networks on resistive device arrays”. In: Frontiers in Neuroscience 14 (2020), p. 103.

Suyog Gupta et al. “Deep learning with limited numerical precision”. In: International conference on machine learning. PMLR. 2015, pp. 1737–1746.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. “Training deep neural networks with low precision multiplications”. In: arXiv preprint arXiv:1412.7024 (2014).
[29] Matthieu Courbariaux et al. “Binarized neural networks: Training deep neural networks with weights and activations constrained to $+1$ or $-1$”. In: arXiv preprint arXiv:1602.02830 (2016).

[30] Lyes Khacef et al. “Self-organizing neurons: toward brain-inspired unsupervised learning”. In: 2019 International Joint Conference on Neural Networks (IJCNN). IEEE. 2019, pp. 1–9.

[31] Peter U Diehl and Matthew Cook. “Unsupervised learning of digit recognition using spike-timing-dependent plasticity”. In: Frontiers in computational neuroscience 9 (2015), p. 99.