Interactive continual learning for robots: a neuromorphic approach

Elvin Hajizada
elvin.hajizada@intel.com
Neuromorphic Computing Lab, Intel Labs
Chair of Cognitive Systems, Technical University of Munich
Munich, Germany

Patrick Berggold
patrick.berggold@tum.de
Chair of Computational Modeling and Simulation, Technical University of Munich
Munich, Germany

Massimiliano Iacono
massimiliano.iacono@iit.it
Event-Driven Perception for Robotics group, Istituto Italiano di Tecnologia
Genova, Italy

Arren Glover
arren.glover@iit.it
Event-Driven Perception for Robotics group, Istituto Italiano di Tecnologia
Genova, Italy

Yulia Sandamirskaya
yulia.sandamirskaya@intel.com
Neuromorphic Computing Lab, Intel Labs
Munich, Germany

ABSTRACT

Intelligent robots need to recognize objects in their environment. This task is conceptually different from the typical image classification task in computer vision. Robots need to recognize particular object instances, not classes of objects, which makes these tasks simpler. However, these instances need to be recognized under different viewing angles, poses, and lighting conditions reliably. Moreover, for many application, robots need the capability to learn new objects quickly, e.g., in an interactive session with the user, and adapt object representations if conditions change and mistakes are made. This scenario creates a demand for object learning that (1) is continual, i.e. new objects can be added at any time without causing forgetting, (2) requires a small amount of data that can be acquired in a short interactive session with the user, and (3) stays open to later adaptation. Deep neural networks trained with slow gradient-based backpropagation, despite of their excellent performance on image processing tasks, are not well-suited for interactive robotic learning tasks. We thus explore smaller neural architectures and increase autonomy of the learning process by a neuronal state machine (NSM). The NSM regulates learning in the network and enables continual adaptation of the learned object prototypes. We implement this model as a spiking neural network in Intel's neuromorphic research chip Loihi and test it on a custom event-based camera dataset generated in a simulated 3D environment. Our spiking neuronal network uses simple feature extraction layers and a single plastic layer that stores visual patterns using online, on-chip local learning rules. This network reaches 96.55±2.02% of testing accuracy on sets of 8 3D objects with 8 different views per object in interactive, on-demand learning experiments; it demonstrates up to x300 energy efficiency and better or on par latency compared to other online learning methods. This work contributes to neuronal-network based machine learning for robots with a small power footprint and interactive learning capability.

CCS CONCEPTS

• **Computing methodologies** → **Computer vision**; **Vision for robotics**; **Scene understanding**; *Supervised learning*; **Online learning settings**; *Instance-based learning*.

KEYWORDS

Neuromorphic computing, spiking neuronal networks, event-based vision, continual learning, Hebbian-learning, on-chip learning, local learning, online learning, robot vision, prototype-based approaches, interactive learning

ACM Reference Format:
Elvin Hajizada, Patrick Berggold, Massimiliano Iacono, Arren Glover, and Yulia Sandamirskaya. 2022. Interactive continual learning for robots: a neuromorphic approach. In *International Conference on Neuromorphic Systems (ICONS 2022), July 27–29, 2022, Knoxville, TN, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3546790.3546791

1 INTRODUCTION

When a human child learns a new object, he or she takes a look, maybe turns the object around, asks what it is, and – some magic happens – they can recognize it again in all kinds of settings and conditions, without compromising the ability to recognize fifteen other objects learned that day [32]. For our future assistive or manufacturing robots, we'd wish a similar capability. As a new task arises, we would like to show the robot a new object to learn, let it take a good look, maybe from a couple of sides, and then be able to detect this object and distinguish it from a dozen other objects relevant today. Modern AI methods that use deep neural networks excel on many visual learning tasks [26, 31, 44, 53, 63, 64]. However, these "monolithic" DNNs are not well suited for our child-like learning scenario: training them requires a lot of well-prepared data, and

ICONS 2022, July 27–29, 2022, Knoxville, TN, USA

ACM ISBN 978-1-4503-9789-6/22/07…$15.00
https://doi.org/10.1145/3546790.3546791

trying to add new objects to the list requires careful retraining with all data to avoid catastrophic forgetting [20, 24, 43, 52, 73].

To enable continual learning in DNNs, there is a broad variety of methods [46] that improve their applicability in dynamic task setting, e.g. using data replay [22, 52, 58, 66], model growing [55], regularization methods [28], uncertainty-based sampling strategies [13], smart distribution of resources in a neural network [61], and other ways to counteract catastrophic forgetting when new inputs do not match the learned distributions [11, 14, 40]. These methods are typically tested in image recognition benchmarks, which are substantially different from robotic tasks [35]. Data collection and labeling in the real world, limited memory and computational resources, instability of the learning algorithms [34] are some of the challenges that make current methods difficult to apply in robots [35]. As a result, we are still lacking an efficient and elegant solution to object learning for robots [47].

Outside of the deep learning domain, there exists another set of approaches that address incremental object learning in robotics, e.g., for mobile robots [29, 37, 39]. These methods do not use deep neural networks, but generally rely on bio-inspired feature extractors (e.g. HMAX-like hierarchical features) [29] or feature representations like color histograms [37], or composed receptive field histograms (CRFH) [39]. The incremental online learning is implemented by a training algorithm on top of these extracted features. Such methods include incremental SVM (ISVM) [5, 39], naive Bayes [41], passive-aggresive classifier [7], incremental learning vector quantization (ILVQ) [37], growing neural gas (NG) [16] or even simple stochastic gradient descent (SGD).

In this context, prototype-based approaches are promising candidates for continual object learning, because these methods store object representations more locally, and model capacity can be adjusted by incorporating more reference vectors to learn new objects [6, 37, 38]. Recently, there has been a growing interest in combining deep learning methods with prototype-based approaches to achieve incremental few-show learning: the core idea is that there exists an embedding in which the points cluster around a single prototype representation for each class [6, 50, 60, 71].

In this paper, drawing inspiration from such prototype-based approaches, we propose a new spiking neural network architecture that does not rely on backpropagation in a deep neural structure and instead localizes learning to a single layer of a multi-layered network. This allows us to fully deploy the network on neuromorphic hardware, making use of its on-chip local synaptic plasticity to enable online, continual learning. The network uses simple pre-wired bio-inspired feature extraction and a plastic read-out layer, achieving fast, single-shot memory formation. Learning in our neural network is controlled by a neuronal state machine – a circuit of interconnected neurons that observes the activity in the input and output layers of the network and "decides" if a weight update is required, a label needs to be requested, or a new readout neuron has to be recruited. Thanks to this control mechanism, we do not need to separate learning and recognition phases, weight adaptation can be turned on as needed, based on the user feedback or self-detected errors.

We tested our model on online learning of representations of a small number of object instances presented in different poses or viewing angles. We use a robotic simulator Gazebo to generate several views of the 3D objects, simulate an event-based camera that samples these views with "microsaccadic" movement, and show how our model can learn to discriminate the objects in a simulated interactive setting. We believe this type of continual learning model together with the advantages of its neuromorphic implementation (e.g. energy efficiency, good scalability, and online learning) will lead to practical solutions for robotic object learning systems. We demonstrate an implementation of such a system on neuromorphic hardware.

2 METHODS

2.1 Neuromorphic hardware

In this work, we implement the neural network architecture on Intel's neuromorphic research chip Loihi [8]. As the other neuromorphic hardware devices, Loihi harnessed insights from biological neural systems to build electronic devices that realize biological computing principles efficiently [56]. Loihi has shown orders of magnitude improvements in terms of low power consumption and fast processing speed on a range of AI tasks [10]. In addition, the on-chip bio-inspired synaptic plasticity on Loihi supports on-chip continual learning.

Each Loihi chip contains 128 neuronal cores, implementing 128K spiking leaky integrate-and-fire (LIF) neurons and 128M synaptic connections, characterized by their weight, w_{ij}, and delay, d_{ij}. The on-chip learning engine updates the weights of the learning layer based on the synaptic plasticity equations that define the learning rule [8]. The learning rule can update weights based on label signals obtained from an expert (e.g. human) while the system is in use. Our network uses approximately 18K neurons and 0.5M synapses on a single Loihi chip.

In our experiments we combine Loihi with an event-based camera, the Dynamic Vision Sensor (DVS) [2, 36], which in turn achieves low-power, low-latency, and high dynamic range in visual sensing [17] and matches the event-based nature of the processing in SNNs. The DVS only produces output if there is a change in the visual field. Since in our experiments we deal with potentially static objects, we use small camera movements, "microsaccades" to generate events in our experiments.

Different methods can be used to develop algorithms, or SNN architectures, for neuromorphic hardware. Conversion of the conventional deep convolutional networks to spiking neural networks (SNNs) [49] and surrogate gradient methods [59] allow us to use the principles of deep learning to create networks that solve pattern classification tasks. Several attempts have been made to enable online learning in such deep networks, e.g., by making the last layer of the network plastic to learn new patterns using features extracted by the early layers of the pre-trained network [23, 62]. We pursue a similar route in our work, using the plastic layer to learn object prototypes in an online fashion, controlled by a neuronal state machine.

2.2 Spiking Neural Network Model

The spiking neuronal network used for object learning in this work consists of three fixed feature extracting layers, a learning layer for object instances and views, and a neuronal circuit that implements a Neuronal State Machine (NSM) as shown in Fig. 1. All layers and

the NSM consist of spiking leaky-integrate-and-fire (LIF) neurons running on Intel's neuromorphic research chip Loihi [8].

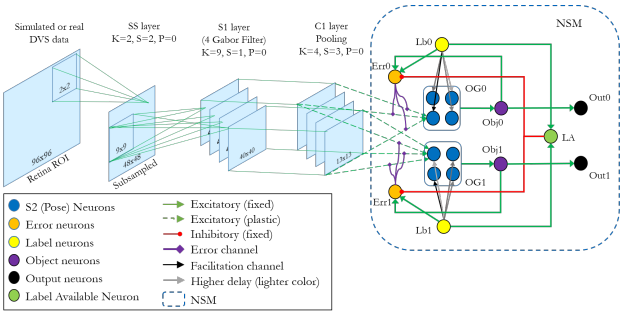


Figure 1: Overview of the proposed architecture. The feature extraction part of the architecture consists of subsampling (SS), S1, and C1 layers, which together generate the feature space of the input. The following plastic layer S2 (dark blue neurons) is organized into object groups (OG) and learning objects based on the extracted features and expert-provided labels. The learning is controlled by the Neural State Machine (NSM) through the error and label neurons and the dendritic compartments of S2 neurons. The network's inference decision is read out from output neurons

2.2.1 Visual Feature Extraction. The feature extraction part of the architecture consists of subsampling (SS), S1 and C1 layers, inspired by the previous work on HMAX architectures [27, 42, 45, 57].

Subsampling Layer. The subsampling layer (SS) uses a convolutional kernel size of 2×2. With an appropriate membrane voltage threshold and a time constant of voltage decay (Table 1), this layer filters out the noise events in the input based on the assumption that noisy events are spatiotemporally less persistent. We set a very short time constant for LIF neurons in the SS layer so that they can spike only once for every microsaccade [27].

S1 Layer: Gabor Filtering. The second layer, S1, is similar to the simple-cells layer of the cortically inspired visual pipeline [42, 45]. The S1 is an event-based convolutional layer, responsible for extracting the edges from the SS layer with four oriented Gabor filters (0, 45, 90, and 135 degrees). There are, respectively, four 2D neuronal oriented-edge feature maps in the S1 layer. The Gabor filters have wavelength 4, standard deviation 2, and aspect ratio 0.5; they are applied with a convolutional window size of 9×9 and stride of 1. The output of the S1 forms four binary (spike/no spike) feature maps. Similar to the SS layer, the S1 neurons can spike only once per input instance; they indicate if an edge with the specific orientation is present in their receptive field. Following [45], an offset of $\frac{\pi}{8}$ is applied to Gabor filters.

C1 Layer: Pooling. The S1 layer is followed by the first complex cell layer C1, also forming four neuronal grids. The neurons in the C1 layer perform local max pooling over a window of 4×4 with stride 3. A C1 neuron will spike if any of the S1 neurons in its receptive field fires. This layer adds local position invariance to the oriented edges. Each C1 neuron can spike only once per instance, thus, its output per microsaccade is binary and forms the extracted feature map to be associated with different views of objects in the next layer.

2.2.2 The Continual Classifier: Plastic Layer and Learning Process.

S2 layer: Prototypes. The next layer, S2, is fully connected from C1 with plastic synapses, all of which have zero initial weights. S2 neurons are divided into n_{Obj} Object Groups (OG). Each OG is pre-allocated for an object, while each neuron in this OG learns a combination of complex features (C1 activation), i.e. a prototype, for a specific view of this object that is sufficiently different from previously learned views. When S2 neurons fire, their synaptic weights change based on the timing of pre- and post-synaptic spikes and activity of the Label and Error neurons (Fig. 1), thus updating the complex features (prototypes) that the S2 neurons are detecting. Particularly, the Label neurons provide the S2 neurons with a supervision signal from a user (e.g. human), while the Error neurons calculate and propagate error signals to C1→S2 synapses (i.e., prototypes) comparing this user-provided label to the network's prediction, all happening while the system is in use.

Learning Process and Local Learning Rule. The learning process of the neural network consists of three components: (1) **allocation**–learning new objects or novel views of the known objects as new prototypes by allocating untaken S2 neurons; (2) **merging**–updating these prototypes by incorporating new object views into them if these are similar to stored prototypes, hence learning gradual conceptual drifts; (3) **punishment**–weakening the active synapses of any prototype that has incorrectly responded. All these processes are automatic: there is no need to turn on and off the learning or explicitly state if the network should learn or infer. Specifically, all three learning components can proceed anytime the system is in use, but only if a label signal is provided. Otherwise, the network makes an inference, i.e. makes its best guess based on the similarity between the provided instance and the stored prototypes. Notably, the similarity measure in our architecture is the activation levels of the S2 neurons (simple dot product between binary feature spike from C1 and non-negative integer weights of C1→S2 synapses) that is thresholded to generate output spike by the S2 neurons. As neither the feature input nor the weights are normalized, the similarity is not normalized either. This limitation of the current architecture is discussed in the results and discussion sections.

The neuromorphic implementation of this learning procedure consists of a learning rule for the plastic layer of synapses and a neural state machine (NSM) to calculate and provide the necessary signals to this layer. The synaptic plasticity rule is mathematically described by the following equations:

$$\Delta w(t) = a^{+} x_j(t)(y_i(t) - y_{imp})\sigma_i(t) - a^{-} e_{ij}(t)\sigma_e(t), \quad (1)$$

$$\sigma_i(t) = \sum_k \delta(t - t_k^i), \quad (2)$$

$$\frac{d}{dt} e_{ij}(t) = b^{+} x_j(t)\sigma_i(t) - e_{ij}(t)/\tau_e, \quad (3)$$

$$\frac{d}{dt} x_j(t) = x_{imp}\sigma_j(t) - x_j(t)/\tau_x, \quad (4)$$

Table 1: The parameters of the proposed network on Loihi (unitless, footnote on p.10 for SDK details).

Name	v_{th}^{SS}	v_{th}^{S1}	v_{th}^{P1}	v_{th}^{S2}	τ_v^{S2}	d_{ref}^{S2}	t_{start}^{Lb}	t_{dur}^{Lb}	n_{OG}	$n_{S2PerOG}$	n_{steps}
Value	11	40	5	4000	20	3	0	21	8	8	100

$$\frac{d}{dt}y_i(t) = y_{imp}\sigma_i(t) - y_i(t)/\tau_y. \quad (5)$$

Here, $x_j(t)$ and $y_i(t)$ are the pre- and postsynaptic traces of spikes; x_{imp} and y_{imp} is the amplitude of impulse that is added to the respective traces every time the pre- and postsynaptic neurons spike; $\sigma_i(t)$ and $\sigma_e(t)$ are the spike trains of the postsynaptic (S2) neuron and the Error neuron (3rd factor), respectively; e_{ij} is the eligibility trace stored in the tag variable of the synapses and τ_x, τ_y, τ_e are the decay time constants of the corresponding traces; $a^+, a^-, b^+ > 0$ are the positive learning rates.

The first term in Eq. (1) accounts for synaptic potentiation (weight increase), including both allocation (recruitment of new neurons) and merging components. The second term describes the synaptic depression (weight decrease), i.e. punishment. More specifically, the depression component of the Eq. (1) is a 3-factor learning rule [33, 67, 70] and serves to decrease/punish the weights which contributed to an error: if there is an error spike, i.e. $\sigma_e(t) \neq 0$, then all synapses with nonzero presynaptic activity–those that have been recently active–will decrease their weights. For this purpose e_{ij} keeps the trace of presynaptic activity that has preceded postsynaptic activity and decay with the τ_e time constant (Eq. (3)). This so-called eligibility trace [18] is applied to the synapse as negative weight change only when an error spike is present. The error signals are calculated by NSM based on the network's prediction (inference) and the user-provided label, as described in Sec. 2.2.3.

In Loihi 1 chip, each synapse can receive only one broadcasted 3rd factor signal [9], however, in our network we have two such signals: label and error. We chose to utilize this channel to propagate error signals. Hence the potentiation term needs to be based on only pre- and postsynaptic traces. This led us to design the second (potentiating) term as a modified Hebbian learning rule: it includes the additional multiplicative term $(y_i(t) - y_{imp})$ which modulates the positive weight change based on the amount of the postsynaptic activation. Practically, this implements a version of the facilitation learning mechanism [48].

However, this approach brings novel challenges. First, as the learning is never "turned off", we need a mechanism to guarantee that during inference the weights are not changed by the learning rule. This is achieved by exploiting the fact that any S2 neuron (postsynaptic in the learning rule) can spike only once per input pattern during inference, because of the single-spike propagation in our network. This fact guarantees that the term $(y_i(t) - y_{imp})$ in Eq. (1) is zero at the time of the first postsynaptic spike during inference, since by definition $y_i(t) = y_{imp}$ after the first spike. This makes the weight update also zero (note that the depression term is zero too because there is no error signal during inference). The rest of the potentiation term $(a^+ x_j(t)\sigma_i(t))$ is classic Hebbian learning: the weight update is proportional to the value of the presynaptic trace, $x_j(t)$, at the time of the postsynaptic spike. Altogether, the network can switch to inference without explicitly turning off the learning on chip, as the learning rule update is zero in the single-spike case of inference.

The second challenge is that we have two types of synaptic potentiation (allocation and merging) that need to be differentiated. This differentiation is done by the facilitation-based learning, which updates the weights depending on the level of the postsynaptic activity. This activity is controlled by the NSM during learning. The initial weights of all S2 neurons are zero. For allocation we want the network to learn the presented pattern as a new prototype. This requires a large positive weight change on the corresponding synapses. On the other hand, for merging we don't want to entirely overwrite the existing prototype, rather superimpose an intensity-wise scaled-down copy of the new pattern onto this prototype[1]. Thus merging requires much smaller synaptic potentiation. The different level of potentiation is achieved by modulating $(y_i(t) - y_{imp})$ term: the larger is the postsynaptic activation $y_i(t)$, the higher is the potentiation. In turn, $y_i(t)$ is controlled by the facilitating label signal: when S2 neurons are facilitated by the label signal, they are artificially forced to spike more than once per instance. This makes $y_i(t) - y_{imp} > 0$ after the second postsynaptic spike, hence potentiating the weights according to the amount of the facilitation, and making the difference between allocation and merging. The level of the provided facilitation is controlled by NSM, as described in Sec. 2.2.3.

Overall, our approach can be seen as the sequential clustering of patterns that present different views of an object. Allocating new S2 neurons is similar to defining a new cluster with the new pattern as the center while merging shifts the center of the cluster towards the sum of the prototype and the new pattern. Synaptic depression triggered by an error signal pushes the cluster center towards a vector that is orthogonal to the presented pattern. This way the SNN achieves supervised, dynamically evolving clustering of the presented patterns, leading to fast learning and continual adaptation using local on-chip learning rules.

2.2.3 The Neural State Machine. The main contribution of this work is to show how learning and inference can be fully automated in the neuronal network. In particular how a spiking neuronal network on a chip, operating in an interactive scenario, can use the concept of the neuronal state machine (NSM) for this purpose. Our NSM is a small interconnected net of neurons that receive labels from a user, detects different states of the overall SNN, and triggers different computing stages, reparametrizing parts of the network on the fly. It controls the inference and learning phases through four main functions: allocating new S2 neurons; detecting representational drift and updating the stored patterns accordingly (merging);

[1]The assumption here is an interactive scenario: the expert does not provide a label for an object instance that is already correctly inferred by the network

calculating error signals; generating other state outputs representing its knowledge of the input (both the visual and the label input). Different parts of the NSM are highlighted in the subfigures of Fig. 2 and are presented in detail below.

S2 Neuron Allocation. In our network, the facilitation signal (introduced in Sec. 2.2.2) is provided by the label neurons' spikes through different dendritic compartments ("input channels") of the S2 neurons (see Fig. 2b). The S2 neurons have a *Soma* compartment (dark blue circle) which receives plastic synapses from the C1 layer and four different dendritic compartments ($D1 - D4$). These dendrites propagate label signals from a Label neuron Lb to the *Soma* and regulate allocation and merging.

In the case of the allocation, the direct $D1$-pathway is used: each S2 neuron in a group receives the label signal at a slightly different point in time because the corresponding $Lb \rightarrow D1$ synapses have different delays. This makes one by one allocation of the S2 neurons possible: when a label signal is presented to a specific object group for the first time, the neuron with the smallest D1 delay will be activated and inhibit all other D1 compartments that are part of the same object group. Then the neural processing advance as follows: the synapse from D1 to S2 is plastic and initially strong; through this synapse the user-activated label neuron provides strong excitation to the corresponding S2 neuron for the first time and driving $(y_i(t) - y_{imp})$ term to high values (see sec. 2.2.2); this increases the weights for the active synapses, hence learning the pattern. Subsequently, this D1→S2 synapse is fully depressed, i.e. cut to avoid reallocation in the later trials, finally completing the allocation of this S2 neuron. The other S2 neurons in the same group are allocated one by one in the same way.

Merging. For the subsequent patterns presented with the same label, we want the network to check first if these are similar to any stored patterns, before deciding to allocate a new S2 neuron. This brings us to the case of merging which requires the automatic superimposition of a labeled pattern that is similar to one of the stored prototypes. Here we implement this by combining two mechanisms: (1) detection of similarity between the presented pattern and the stored prototypes and (2) moderate facilitation of the S2 neuron with the highest similarity to update this prototype. The first mechanism is realized by the following pathway: the provided label signal boosts activation of all S2 neurons by some amount through $D2$ compartment; if one of the allocated S2 neurons fires, this is considered as detection of high similarity and subsequently this neuron blocks any new neuron allocation. The same spike together with the label spike generates a positive feedback loop through D3 and D4 compartments ($Soma \rightarrow D4 \rightarrow D3 \rightarrow Soma$) and force the *Soma* to spike more, hence potentiating active C1→S2 synapses; this potentiation is the merging of the stored pattern and the input. However, this is a weaker potentiation compared to the allocation case, and hence only adapts the weights slowly to track any slow representational drift.

Error Signal Calculation. As we introduced in sec. 2.2.2, during the learning process, spikes of the error neurons provide the third factor for the learning rule. For each object neuron there is one Error neuron which signals that this object neuron spike does not correspond to the label neuron spike. This is computed as follows

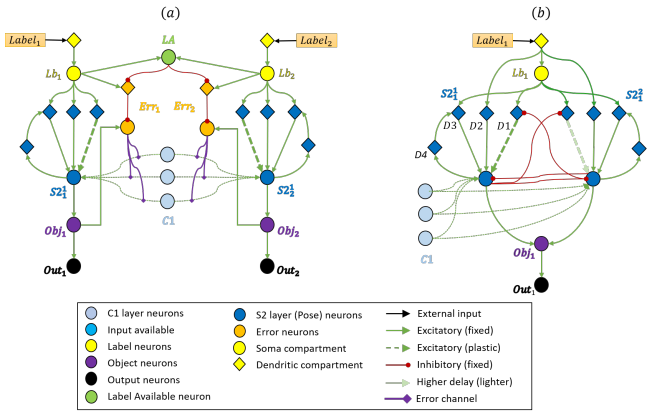


Figure 2: Neural State Machine (NSM). (a) Controlling the autonomous learning process. (b) Error signal calculation during the training instances for two objects. (c) Autonomous allocation and merging of view neurons in S2 for two S2 neurons from one object group. See text for details.

(Fig. 2a): when a label is provided, it excites the inhibitory dendritic compartment (orange rhombus) of the respective Error neuron (hence inhibiting it) and also activates the "label available" (LA) neuron. In turn, the LA neuron inhibits the dendritic compartments of all Error neurons, except the one that is excited by the active label neuron. Thus, if a label is provided, then all Error neurons are disinhibited, except the one that corresponds to the label. The error neurons that are disinhibited can respond to an incoming object neuron spike. Each such spike generates an error spike which signals that the predicted object does not correspond to the provided label.

Auxiliary Neural State Outputs. To enable the interactive scenario, the object recognition network in addition to its classification decisions detects and signals to the rest of the network, the other behavioral modules (e.g. attention, speech), and the user what else it "knows" about the currently presented the object instance and its label. For this purpose, we implemented six additional neuronal groups as the part of the neuronal state machine (not shown in the figure): "input available" (IA) neuron indicates if the network has received any visual input; "known object" (KO) and "unknown object" (UkO) neurons signal if the network "knows" the object, i.e. has strong confidence about object's class; "known label" (KLb) and the "unknown label" (UkLb) neurons report if the network has previously "heard" a specific expert-provided label; "learning finished" (LF) neuron signals that the learning process of the current object instance is finalized. However, the use case of these neurons is outside the scope of the experiments and the setup presented in this paper. These neurons can be used in a closed-loop learning scenario to control the activation of the robot's behaviors: shifting attention, moving the sensor to collect events, or asking the user for a label.

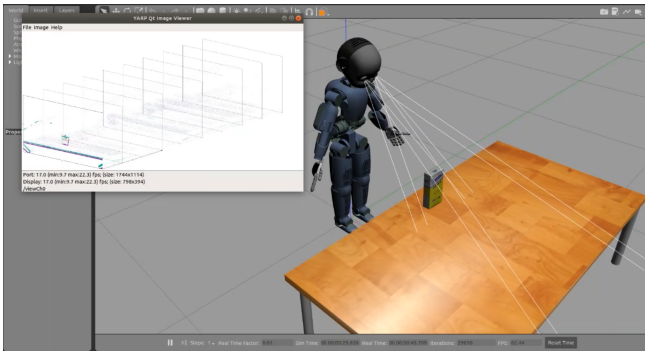


Figure 3: Experimental setup for data collection in Gazebo simulation environment.

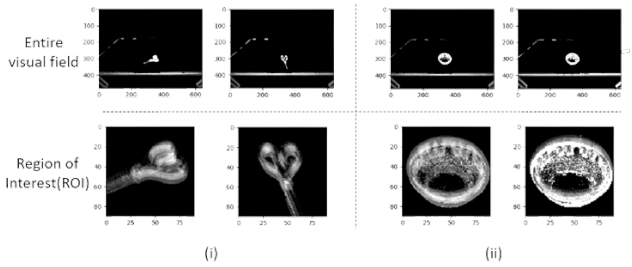


Figure 4: The examples of collected visual output: the entire visual field (camera view) and the attended region-of-interest (ROI) for two different objects–scissors and a bowl.

3 EXPERIMENTS

3.1 Simulated Experimental Setup for Data Collection

To test our neuronal network for interactive continual object learning, we used a simulated robotic environment in the Gazebo simulator (Fig. 3) running the model of the iCub robot [65]. In Gazebo, we placed the simulated iCub robot in front of a table and rendered 3D objects from the YCB data base [3, 4] on the table to create a visual scene for iCub.

We put one of 19 different daily objects from the YCB data base on a predefined location on the table, to which the iCub's gaze is directed. We used an event-based camera simulator eSim [51] to generate events based on the simulated iCub's camera videos. Since in our object learning scenario, the robot gazes at the static objects, we need to move the robot's camera to generate the events and "see" the objects. These eye movements are inspired by the small fixational eye movements in humans called microsaccades (MS) [30]. Some examples of the generated output are shown in Fig. 4. Note that we don't exploit any feature-extracting properties of MS in this work, but would like to point out this opportunity for future research.

Each object is placed at one of 20 different angles to simulate different viewing angles on the objects. For each object placement, the robot gazes at the object and performs 5 microsaccades. We repeat this process for all objects and rotations. In the end, we collected event stream data recordings for 20 rotations of 19 objects, leading to 380 recordings, where each recording contains 5 microsaccades. The described simulator setup can be used to generate new samples to benchmark the model and to simulate interactive learning on a real robot. We will release the current dataset upon paper publication for benchmarking by other researchers.

3.2 Preprocessing of the Event Stream.

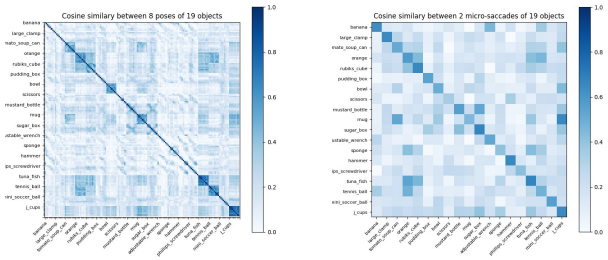
In our implementation, the event stream generated by microsaccades is first preprocessed on a CPU. Each recording is divided into multiple microsaccadic "chunks" of 100 ms (one chunk = one microsaccade) and only a region of interest (ROI) that is centered on the object (Fig. 4) is extracted[2]. This preprocessing step is a placeholder for a bottom-up attention mechanism for objects in the scene, e.g. [68, 69]. In future work, the ROI will be selected by a bottom-up saliency and visual search-driven top-down attention module [12, 15, 19, 21, 25, 54].

3.3 Evaluation Framework

We evaluated the interactive continual learning capabilities of our architecture in a simulated interaction with a user. In each experiment, 8 objects are randomly chosen among 19 recorded ones. For all chosen objects the first 8 rotations are selected, accounting to overall 64 recordings. Each recording is separated into two groups: out of 5, the first three microsaccades are used for learning, while we tested the learned representations on the next two. To train the network the "user" presents the training microsaccades of all 64 recordings (8 rotations, 8 objects) one by one in a random sequence. After each input, the user observes the output of the network and compares it to the actual label and if the prediction is incorrect, it presents the same input again together with the label signal, otherwise, it proceeds to the next input. In other words, we simulate an on-demand online learning process.

Each of such learning sessions is followed by a testing session, to demonstrate what the network has learned so far. The test microsaccades for all objects and rotations are presented to the network sequentially and the inference outputs are observed by the user. In these sessions no label signal is provided, rather the user measures the accuracy for the test microsaccades of the already learned objects. The on-demand learning and testing sessions are interleaved to measure the evolution of the learning process. We present the results of these experiments in the following section together with an analysis of the Gabor filter generated latent space and a comparison to the other online learning methods in terms of accuracy, speed, and energy consumption.

[2]The ROI size was set 96×96 pixels and chosen manually here as a bounding box of the object in the scene. The number of events in each chunk is variable and all events in a chunk are injected into the network on chip in one (algorithmic) time step. The network then processes this input in 100 (algorithmic) time steps. As we will see in Sec. 4 the processing of one instance (i.e. 100-time steps) takes only 2.6ms, which is much faster than the microsaccade itself (100 ms).



(a) (b)

Figure 5: Cosine similarity between the C1 layer outputs (the extracted features from different objects): among (a) 8 rotations of all objects, (b) two microsaccades of all objects (the same rotation)

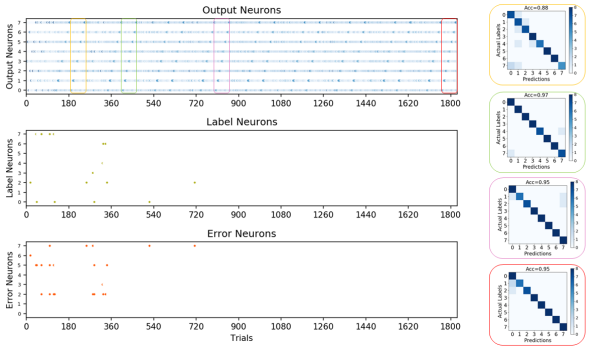


Figure 6: Single iteration of a continual learning trial showing spikes of output, label, and error neurons. For several testing sessions, the confusion matrices are shown on the right. After the first iteration over the training set (all three training MS), there are almost no errors and hence no label requests. The objects in this trial: 0-large clamp, 1-tennis ball, 2-tuna fish box, 3-banana, 4-scissors, 5-sponge, 6-screwdriver, 7-can.

4 RESULTS

4.1 Discrimination capacity of the feature extraction layers

Since we use a shallow and simplistic feature extraction, we present an analysis of the latent space, created by the S1 and C1 layers for different batches of 8 objects with 8 rotation each, sampled by 5 different microsaccades (Fig. 5), to evaluate the complexity of the continual learning task, solved by the plastic layer of the SNN.

To explore the extracted feature representations, we computed cosine similarity matrices for all combinations of the different objects, their views, and microsaccades. Accordingly, Fig. 5a displays the similarity between different poses of all objects, demonstrating that while some objects look similar under different views (darker blue blocks around the diagonal), other objects are not similar in different views and may overlap with views of the other objects (off-diagonal dark blue patches). The similarity matrix for two different microsaccades for the same view of objects in Fig. 5b furthermore reveals that different microsaccades may sample the object's appearance differently, making the classification task harder. This exploration of the latent space indicates the limits for the best possible accuracy of the subsequent classifier.

4.2 Continual learning of patterns

We perform our main experiment as described in Sec. 3.3. In all tests presented here, the following setting is used: 8 objects and 8 poses per object; for each of these 64 object instances 5 microsaccades (MS) are recorded, summing up to 320 microsaccadic chunks. The first three MSs are used during on-demand learning, while test phases measure performance on the two unseen MSs. This process is repeated for 3 epochs. The object-pose instances are randomly shuffled during training. We limited the number of the classes for our classifier to 8 objects to stay in the computational limits of a single chip, however the objects are chosen randomly from the 19 recorded objects for each iteration of our experiment.

Fig. 6 shows a single run of the experiment where on-demand learning sessions interleaved with testing phases. The three time series plots show spikes of the Output, Label, and Error neurons. Through the testing sessions, we assess the performance of the network at different points in time, namely after each on-demand learning session: all poses of all objects are presented one by one, and responses are stored in a confusion matrix. Several snapshots of the confusion matrix are shown on the right-hand side of the figure. We can see that already after the first learning epoch, the confusions are rare (<20%) and the accuracy of the SNN further improves, reaching 96.55±2.02% after three learning epochs. This result is obtained from 12 repetition of the experiment with different sets of 8 objects chosen randomly from the 19 recorded objects.

Notably, in these tests, we use the value of the internal state variable of LIF neurons to force a decision for an object, which increases confusion. Alternatively, an output activation threshold can be introduced and the "unknown object" neuron would be used to signal low confidence in the decision, leading to practically zero confusion errors after a learning episode.

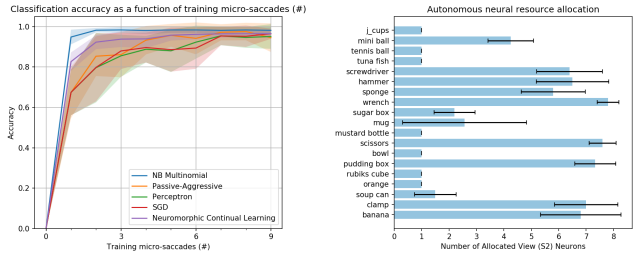
Further analysis of the Fig. 6 reveals some important observations. First and foremost, the sparse spiking of Label and Error neurons points at sparse weight update, as a result of the on-demand learning process. If the object is inferred correctly, no label is provided and no weight update happens. Only if a wrong object is selected and this mistake is detected by the "user", the label is provided. Then this error is also detected by the NSM, which depresses the responsible weights (see Sec. 2.2.2 and 2.2.3 for more details). Therefore, the distance between the correct and wrong classes in the feature space is increased. The network makes fewer and fewer mistakes as the online training proceeds.

Another crucial result from these experiments is the resource efficiency of the proposed architecture. Fig. 7b shows the number of S2 neurons allocated in different object groups during the continual learning process. One can observe that more neurons are allocated for objects that look different under different viewpoints, while objects with rotational symmetries require fewer prototype (S2) neurons (e.g. bowl, Rubik's cube, or soup can). This demonstrates autonomous learning of the efficient representations, whose complexity depends on the complexity of the object. Even though in the

Table 2: Processing/execution time and energy consumption benchmarking results together with classification accuracy. The experiments are conducted for a set of 8 objects with 8 poses, 3 training, and 2 testing microsaccades, comparing our method (Loihi Continual Classifier) and online linear SVM (SGD), Perceptron, Passive-Aggressive classifier, and Naive Bayesian multinomial approach. We report ET (execution time) and energy consumption per inference and learning instance for all methods. Notably, we achieve up to x300 better energy per learning an instance (and up to x150 for inference), while showing the best execution time for learning an instance and being on par with other methods in inference time

		Execution time (ms)*		Energy consumption (mJ)*	
Learning method	Accuracy(%)	Inference	Learning	Inference	Learning
NBM [41]	98.2 ± 1.7	2.5	2.7	98	107
Passive-Aggressive [7]	94.5 ± 7.1	2.4	4.0	97	163
Perceptron	95.1 ± 6.1	2.4	4.1	98	164
SGD [72]	96.5 ± 2.5	2.4	4.0	99	163
Loihi Continual classifier	96.6 ± 2.0	2.6	2.6	0.6	0.6

*Numbers are reported per learning and inference instance



(a) (b)

Figure 7: (a) Classification accuracy for a set of 8 objects with 8 poses, 3 training and 2 testing microsaccades: comparison between our method and online linear SVM (SGD), Perceptron, Passive-Aggresive classifier, and Naive Bayesian multinomial approach. The accuracy is recorded during each test phase, performed after each on-demand learning session (total of 9 test phases). These 9 phases are divided into 3 epochs, each featuring learning over 3 learning MSs in sequence. (b) Number of the pose neurons recruited for different objects in the learning process across 12 different learning runs.

current network the maximum number of pre-allocated neurons per object is fixed, in the next iteration of the architecture we aim to implement neuron allocation from a shared pool to further improve the resource efficiency.

4.3 Comparison to non-neuromorphic approaches

As a next step, we compared the continual classifier of our architecture (S2 prototype layer + NSM) to other online learning methods in the same test setting. Fig. 7a compares the test accuracy for different online learning algorithms for 8 objects and 8 poses. The non-neuromorphic algorithms used for comparison are classification methods and applied directly to the extracted feature vectors of the C1 layer. These classifiers were trained with a mini-batch size of 1, so only a single sample is shown each time (online training) as in our original experiment. The mean accuracy is shown for 12 iterations for each method, the shaded regions are standard deviations of the test accuracy. Our classifier shows a smaller variance in accuracy and converges faster than online linear SVM (Stochastic Gradient Descent with hinge loss [72]), the Perceptron, and a Passive-Aggresive [7] classifiers. Even though the Naive Bayesian classifier [41] achieves slightly higher accuracy, the subsequent processing speed and energy consumption benchmarking reveals the other advantageous sides of our approach.

In this regard, we isolated our continual classifier of the object recognition network and benchmarked it against these conventional online classifier methods running on CPU, in terms of processing speed and energy consumption both for learning and inference (Table 2). Results demonstrate up to x300 better energy per learning instance (and up to x150 for inference) while showing the best execution time for learning an instance and being on par with other methods in inference time. Note, each training and testing instance takes 100 timesteps in Loihi 1, hence reported execution times are per 100 timesteps. The latency and energy numbers for inference and learning are the same in our architecture. Clearly, the effect of learning on latency and energy is negligible because the network uses fast-acting learning rules that are precisely gated in time: less than 0.4% of the weights in the network need to be updated per 100 timesteps. The requirement for 100 timesteps is an algorithmic constraint, which will be relaxed in future work, featuring the next implementation of continual classifier in Loihi 2 hardware[3].

5 DISCUSSION AND CONCLUSION

In this work we present a new approach to neural network-based object learning, specifically targeting applications in robotics where learning new objects after deployment may be required. The learning process can unfold autonomously, in interaction with the user. Objects' views can be learned quickly by localizing learning to a single layer of plastic synapses and ensuring that the complexity required to store different object views is accounted for by recruiting

[3]All experiments run on a machine with Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz x8, 32GB RAM running Ubuntu 20.04.2 LTS, python 3.8.10, and v. 1.0.0 of the Intel NxSDK in Loihi 1 hardware. All performance measurements are based on testing as of October 2021 and may not reflect all publicly available security updates. Results may vary.

new local-view neurons, i.e. prototypes. We present a simple, small-scale example here that can be scaled and made robust to variations in the visual appearance of objects, e.g. increasing scale and shift invariance [1] and using richer feature representations. We reached 96.55±2.02% accuracy in our interactive, on-demand learning experiments and demonstrated up to x300 energy efficiency and better or on par latency compared to other online learning methods, despite the current architectural and algorithmic limitations.

The absence of the weight normalization or the explicit long-term voltage threshold adaptation is one limitation of the current architecture: the unnormalized dot product is an unreliable similarity measure. Additionally, the feature extraction layer is very simple, having only one layer of Gabor filters. The dataset that is generated and used for the learning and testing is also small in the context of modern AI literature, however as the explored domain is novel in many aspects, we believe this dataset is a good start and the data generation pipeline allows future expansion. We will need to test our algorithm in the real-world with actual robots and many more objects to understand scaling of our algorithm.

Overall, this work contributes to learning approaches in robotics that work in interactive settings autonomously, while allowing us to build on the success achieved in deep learning on computer vision benchmarks. Our approach relies on using a robot simulator and can be extended to the physical platform with closed-loop behavior to test the autonomy of the learning process and to demonstrate continual learning. As the next step, we see a lot of potential in comparing state-of-art continual learning approaches to our architecture in this kind of close loop interactive learning setting.

ACKNOWLEDGMENTS

Intel Labs, Neuromorphic Computing Lab has provided the access to Loihi hardware and support in terms of software. We also thank Prof. Gordon Cheng for his support during this project.

REFERENCES

[1] Sandro Baumgartner, Alpha Renner, Raphaela Kreiser, Dongchen Liang, Giacomo Indiveri, and Yulia Sandamirskaya. 2020. Visual Pattern Recognition with on On-chip Learning: towards a Fully Neuromorphic Approach. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.

[2] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. 2014. A 240× 180 130 db 3 μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits* 49, 10 (2014), 2333–2341.

[3] Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. 2017. Yale-CMU-Berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research* 36, 3 (2017), 261–268.

[4] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M. Dollar. 2015. Benchmarking in Manipulation Research: Using the Yale-CMU-Berkeley Object and Model Set. *IEEE Robotics Automation Magazine* 22, 3 (2015), 36–52. https://doi.org/10.1109/MRA.2015.2448951

[5] Gert Cauwenberghs and Tomaso Poggio. 2000. Incremental and decremental support vector machine learning. *Advances in neural information processing systems* 13 (2000).

[6] Kuilin Chen and Chi-Guhn Lee. 2020. Incremental few-shot learning via vector quantization in deep embedded space. In *International Conference on Learning Representations*.

[7] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive aggressive algorithms. (2006).

[8] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro* 38, 1 (2018), 82–99.

[9] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Prasad Joshi, Andrew Lines, Andreas Wild, and Hong Wang. 2018. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro* PP (01 2018), 1–1. https://doi.org/10.1109/MM.2018.112130359

[10] Mike Davies, Andreas Wild, Garrick Orchard, Yulia Sandamirskaya, Gabriel A Fonseca Guerra, Prasad Joshi, Philipp Plank, and Sumedh R Risbud. 2021. Advancing neuromorphic computing with Loihi: A survey of results and outlook. *Proc. IEEE* 109, 5 (2021), 911–934.

[11] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[12] Giulia D'Angelo, Adam Perrett, Massimiliano Iacono, Steve Furber, and Chiara Bartolozzi. 2022. Event driven bio-inspired attentive system for the iCub humanoid robot on SpiNNaker. *Neuromorphic Computing and Engineering* 2, 2 (2022), 024008.

[13] Sayna Ebrahimi, Mohamed Elhoseiny, Trevor Darrell, and Marcus Rohrbach. 2019. Uncertainty-guided Continual Learning with Bayesian Neural Networks. (2019), 1–16. arXiv:1906.02425 http://arxiv.org/abs/1906.02425

[14] Sebastian Farquhar and Yarin Gal. 2018. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733* (2018).

[15] Jérémy Fix, Nicolas Rougier, and Frédéric Alexandre. 2011. A dynamic neural field approach to the covert and overt deployment of spatial attention. *Cognitive Computation* 3, 1 (2011), 279–293.

[16] Bernd Fritzke. 1994. A growing neural gas network learns topologies. *Advances in neural information processing systems* 7 (1994).

[17] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Jörg Conradt, Kostas Daniilidis, et al. 2019. Event-based vision: A survey. *arXiv preprint arXiv:1904.08405* (2019).

[18] Wulfram Gerstner, Marco Lehmann, Vasiliki Liakoni, Dane Corneil, and Johanni Brea. 2018. Eligibility traces and plasticity on behavioral time scales: experimental support of neohebbian three-factor learning rules. *Frontiers in neural circuits* 12 (2018), 53.

[19] Suman Ghosh, Giulia D'Angelo, Arren Glover, Massimiliano Iacono, Ernst Niebur, and Chiara Bartolozzi. 2022. Event-driven proto-object based saliency in 3D space to attract a robot's attention. *Scientific reports* 12, 1 (2022), 1–14.

[20] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211* (2013).

[21] Raul Grieben, Jan Tekülve, Stephan KU Zibner, Jonas Lins, Sebastian Schneegans, and Gregor Schöner. 2020. Scene memory and spatial inhibition in visual search. *Attention, Perception, & Psychophysics* (2020), 1–24.

[22] Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. 2020. Remind your neural network to prevent catastrophic forgetting. In *European Conference on Computer Vision*. Springer, 466–483.

[23] Tyler L Hayes and Christopher Kanan. 2020. Lifelong machine learning with deep streaming linear discriminant analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 220–221.

[24] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. 2019. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 831–839.

[25] Massimiliano Iacono, Giulia D'Angelo, Arren Glover, Vadim Tikhanoff, Ernst Niebur, and Chiara Bartolozzi. 2019. Proto-object based saliency for event-driven cameras. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 805–812.

[26] Elia Kaufmann, Antonio Loquercio, Rene Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. 2018. Deep drone racing: Learning agile flight in dynamic environments. In *Conference on Robot Learning*. PMLR, 133–145.

[27] Saeed Reza Kheradpisheh, Mohammad Ganjtabesh, Simon Thorpe, and Timothée Masquelier. 2017. STDP-based spiking deep convolutional neural networks for object recognition. *Neural Networks* 99 (12 2017). https://doi.org/10.1016/j.neunet.2017.12.005

[28] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114, 13 (2017), 3521–3526.

[29] Stephan Kirstein, Heiko Wersing, and Edgar Körner. 2005. Rapid online learning of objects in a biologically motivated recognition architecture. In *Joint Pattern Recognition Symposium*. Springer, 301–308.

[30] Hee-kyoung Ko, Martina Poletti, and Michele Rucci. 2010. Microsaccades precisely relocate gaze in a high visual acuity task. *Nature neuroscience* 13, 12 (2010), 1549–1553.

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012).

[32] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. 2011. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, Vol. 33.

[33] Robert Legenstein, Dejan Pecevski, and Wolfgang Maass. 2008. A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback. *PLoS Comput Biol* 4, 10 (2008), e1000180.

[34] Timothée Lesort, Hugo Caselles-Dupré, Michael Garcia-Ortiz, Andrei Stoian, and David Filliat. 2019. Generative models from the perspective of continual learning. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[35] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. 2020. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information fusion* 58 (2020), 52–68.

[36] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. 2008. A 128× 128 120 dB 1 5μs latency asynchronous temporal contrast vision sensor. *IEEE journal of solid-state circuits* 43, 2 (2008), 566–576.

[37] Viktor Losing, Barbara Hammer, and Heiko Wersing. 2015. Interactive online learning for obstacle classification on a mobile robot. In *2015 international joint conference on neural networks (ijcnn)*. IEEE, 1–8.

[38] Viktor Losing, Barbara Hammer, and Heiko Wersing. 2018. Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing* 275 (2018), 1261–1274.

[39] Jie Luo, Andrzej Pronobis, Barbara Caputo, and Patric Jensfelt. 2007. Incremental learning for place recognition in dynamic environments. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 721–728.

[40] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. 2021. Online Continual Learning in Image Classification: An Empirical Survey. 1 (2021), 1–64. arXiv:2101.10423 http://arxiv.org/abs/2101.10423

[41] CD Manning, P Raghavan, and H Schütze. 2008. Xml retrieval. In *Introduction to Information Retrieval*. Cambridze University Press.

[42] Timothée Masquelier and Simon Thorpe. 2007. Unsupervised Learning of Visual Features through Spike Timing Dependent Plasticity. *PLoS computational biology* 3 (03 2007), e31. https://doi.org/10.1371/journal.pcbi.0030031

[43] Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*. Vol. 24. Elsevier, 109–165.

[44] Stefan Milz, Georg Arbeiter, Christian Witt, Bassam Abdallah, and Senthil Yogamani. 2018. Visual slam for automated driving: Exploring the applications of deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 247–257.

[45] M. Mozafari, S. R. Kheradpisheh, T. Masquelier, A. Nowzari-Dalini, and M. Ganjtabesh. 2018. First-Spike-Based Visual Categorization Using Reward-Modulated STDP. *IEEE Transactions on Neural Networks and Learning Systems* 29, 12 (2018), 6178–6190. https://doi.org/10.1109/TNNLS.2018.2826721

[46] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks* 113 (2019), 54–71.

[47] Giulia Pasquale, Carlo Ciliberto, Francesca Odone, Lorenzo Rosasco, and Lorenzo Natale. 2017. Are we Done with Object Recognition? The iCub robot's Perspective. *Robotics and Autonomous Systems* 112 (09 2017). https://doi.org/10.1016/j.robot.2018.11.001

[48] Alexandre Payeur, Jordan Guerguiev, Friedemann Zenke, Blake Richards, and Richard Naud. 2020. Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. (03 2020). https://doi.org/10.1101/2020.03.30.015511

[49] José Pérez-Carrasco, Bo Zhao, Carmen Serrano, Begoña Acha, Teresa Serrano-Gotarredona, Shoushun Chen, and Bernabé Linares-Barranco. 2013. Mapping from Frame-Driven to Frame-Free Event-Driven Vision Systems by Low-Rate Rate-Coding and Coincidence Processing. Application to Feed Forward ConvNets. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35 (11 2013), 2706 – 2719. https://doi.org/10.1109/TPAMI.2013.71

[50] Hang Qi, Matthew Brown, and David G Lowe. 2018. Low-shot learning with imprinted weights. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5822–5830.

[51] Henri Rebecq, Daniel Gehrig, and Davide Scaramuzza. 2018. ESIM: an open event camera simulator. In *Conference on Robot Learning*. PMLR, 969–982.

[52] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2001–2010.

[53] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. 2022. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*. PMLR, 91–100.

[54] Jonas Ruesch, Manuel Lopes, Alexandre Bernardino, Jonas Hornstein, José Santos-Victor, and Rolf Pfeifer. 2008. Multimodal saliency-based bottom-up attention a framework for the humanoid robot icub. In *2008 IEEE International Conference on Robotics and Automation*. IEEE, 962–967.

[55] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671* (2016).

[56] Catherine D Schuman, Thomas E Potok, Robert M Patton, J Douglas Birdwell, Mark E Dean, Garrett S Rose, and James S Plank. 2017. A survey of neuromorphic computing and neural networks in hardware. *arXiv preprint arXiv:1705.06963* (2017).

[57] Thomas Serre, Lior Wolf, Stanley Bileschi, Maximilian Riesenhuber, and Tomaso Poggio. 2007. Robust object recognition with cortex-like mechanisms. *IEEE transactions on pattern analysis and machine intelligence* 29, 3 (2007), 411–426.

[58] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, Vol. 2017-Decem. 2991–3000. arXiv:1705.08690

[59] Sumit Bam Shrestha and Garrick Orchard. 2018. SLAYER: Spike Layer Error Reassignment in Time. In *NeurIPS*.

[60] Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. *Advances in neural information processing systems* 30 (2017).

[61] Ghada Sokar, Decebal Constantin Mocanu, and Mykola Pechenizkiy. 2021. SpaceNet: Make Free Space for Continual Learning. *Neurocomputing* 439 (2021), 1–11. https://doi.org/10.1016/j.neucom.2021.01.078 arXiv:2007.07617

[62] Kenneth Stewart, Garrick Orchard, Sumit Bam Shrestha, and Emre Neftci. 2020. Online few-shot gesture learning on a neuromorphic processor. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 10, 4 (2020), 512–521.

[63] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*.

[64] Mingxing Tan and Quoc Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*. PMLR, 6105–6114.

[65] Vadim Tikhanoff, Angelo Cangelosi, Paul Fitzpatrick, Giorgio Metta, Lorenzo Natale, and Francesco Nori. 2008. An open-source simulator for cognitive robotics research: the prototype of the iCub humanoid robot simulator. In *Proceedings of the 8th workshop on performance metrics for intelligent systems*. 57–61.

[66] Gido M. van de Ven, Hava T. Siegelmann, and Andreas S Tolias. 2020. Brain-inspired replay for continual learning with artificial neural networks. *Nature Communications* 11, 1 (2020). https://doi.org/10.1038/s41467-020-17866-2

[67] Eleni Vasilaki, Nicolas Frémaux, Robert Urbanczik, Walter Senn, and Wulfram Gerstner. 2009. Spike-based reinforcement learning in continuous state and action space: when policy gradient methods fail. *PLoS Comput Biol* 5, 12 (2009), e1000586.

[68] Roberto A Vazquez, Bernard Girau, and Jean-Charles Quinton. 2011. Visual attention using spiking neural maps. In *The 2011 International Joint Conference on Neural Networks*. IEEE, 2164–2171.

[69] QingXiang Wu, T Martin McGinnity, Liam Maguire, Rongtai Cai, and Meigui Chen. 2013. A visual attention model based on hierarchical spiking neural networks. *Neurocomputing* 116 (2013), 3–12.

[70] Xiaohui Xie and H Sebastian Seung. 2004. Learning in neural networks by reinforcement of irregular spiking. *Physical Review E* 69, 4 (2004), 041909.

[71] Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. 2020. Semantic drift compensation for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6982–6991.

[72] Bianca Zadrozny and Charles Elkan. 2002. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 694–699.

[73] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. 2020. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13208–13217.