Parameter Optimization and Learning in a Spiking Neural Network for UAV Obstacle Avoidance targeting Neuromorphic Processors

Llewyn Salt, David Howard, *Member, IEEE,* Giacomo Indiveri, *Senior Member, IEEE,*
Yulia Sandamirskaya, *Member, IEEE,*

Abstract—The Lobula Giant Movement Detector (LGMD) is an identified neuron of the locust that detects looming objects and triggers the insect's escape responses. Understanding the neural principles and network structure that lead to these fast and robust responses can facilitate the design of efficient obstacle avoidance strategies for robotic applications. Here we present a neuromorphic spiking neural network model of the LGMD driven by the output of a neuromorphic Dynamic Vision Sensor (DVS), which incorporates spiking frequency adaptation and synaptic plasticity mechanisms, and which can be mapped onto existing neuromorphic processor chips. However, as the model has a wide range of parameters, and the mixed signal analogue-digital circuits used to implement the model are affected by variability and noise, it is necessary to optimise the parameters to produce robust and reliable responses. Here we propose to use Differential Evolution (DE) and Bayesian Optimisation (BO) techniques to optimise the parameter space and investigate the use of Self-Adaptive Differential Evolution (SADE) to ameliorate the difficulties of finding appropriate input parameters for the DE technique. We quantify the performance of the methods proposed with a comprehensive comparison of different optimisers applied to the model, and demonstrate the validity of the approach proposed using recordings made from a DVS sensor mounted on a UAV.

Index Terms—Differential Evolution, Bayesian Optimisation, Self-adaptation, STDP, Neuromorphic Engineering

I. INTRODUCTION

State-of-the-art robotic unmanned aerial vehicle (UAV) systems are achieving impressive results for compact and agile flight and manoeuvring [1]. However these systems are typically less power efficient and robust than their natural counterparts (e.g., bees are capable of robust flight, obstacle avoidance, and cognitive capabilities with a neural processing technology that consumes approximately $10\,\mu\text{W}$ of power, and that occupies a volume of less than $1\,\text{mm}^3$). Using nature as inspiration, neuromorphic engineers have attempted to bridge the power-consumption gap through hardware solutions [2]. Recently, a range of different neuromorphic processors has been proposed to allow for the hardware implementation of spiking neural networks (SNNs) [3]–[8]. These mixed-signal analog/digital chips are ultra low power (on the order of mW)

L. Salt is with the School of Information Technology and Electrical Engineering, University of Queensland, Queensland, Australia.

D. Howard is with the Robotics and Autonomous Systems Group in the Cyberphysical System Program, CSIRO, Queensland, Australia.

G. Indiveri and Y. Sandamirskaya are with the Institute of Neuroinformatics, University of Zurich and ETH Zurich, Zurich, Switzerland.

and provide an attractive alternative to current digital hardware used in mobile applications such as robotics.

Another successful neuromorphic example is the recent development of silicon retinas and event-based sensors such as the Dynamic Vision Sensor (DVS) [9], [10]. The DVS operates differently compared to conventional video cameras: instead of integrating light in a pixel array for a period of time and then converting it to an image, it detects local changes in luminance at each pixel and transmits these change events asynchronously as they are detected and with microsecond latency [11]. Compared to standard frame-based cameras, a DVS offers (i) faster response times through asynchronous transmission, (ii) much lower bandwidth, and (iii) no motion blur [12].

A system with sensors and image processing in-situ on the UAV is an essential step for autonomous UAV systems. Typically, high-speed agile manoeuvres such as juggling, pole acrobatics, or flying through thrown hoops, use external motion sensors and high powered CPUs to control the UAVs [13]–[15]; a combination of DVS and Neuromorphic spiking networks provides low-power and high response rates together with the potential for adaptation from the SNNs, and as such are promising technologies for autonomous UAVs.

A model that has shown promise for fast and robust collision avoidance in UAV robotics applications is the locust Lobula Giant Motion Detector (LGMD) [16]–[21]. The LGMD in a locust is capable of responding to an object looming at speeds ranging from $0.3\,\text{m/s}$ to $10\,\text{m/s}$ [22]; our model was tested on stimuli that loomed at rates of 266 pixels per second to 1478 pixels per second. The locust uses the LGMD to escape from predators by detecting whether a stimulus is looming (increasing in size in the field of view) or not [23]. This neuronal looming detection mechanism is robust to translation, which is why it is an ideal candidate for obstacle avoidance. Previous implementations of this model used frame based cameras and simplified neural models for embedded robotic applications [23]–[25]. Salt et al. [26] modified the LGMD model from [16] first mentioned in [27] to use Adaptive Exponential Integrate and Fire (AEIF) neuron equations [28] which model faithfully the behavior of silicon neurons present in hardware neuromorphic processors [3]. The LGMD Neural Network (LGMDNN) was, in particular, modified to make it compatible with the Reconfigurable On-Line Learning Spiking (ROLLS) neuromorphic processor [29]. In this previous work we have presented a proof of concept demonstration that a

LGMDNN network can be used for obstacles avoidance on a UAV. Coupling the LGMDNN with the AEIF neural equations yields 11 user-defined parameters after making simplifying assumptions based on the constraints of the neuromorphic processor. Optimising this parameter space is challenging as it contains complex inter-dependencies. Moreover, these parameters are shared by neurons and synapses in different parts of the structured neuronal network and thus have influence on the overall performance of the network that does not lend itself to any simple description of the role of each parameter. Here we demonstrate the method by which the appropriate parameters were found in [26] and show the extension of this work to incorporate synaptic plasticity and neural spike-frequency adaptation mechanisms.

Identifying acceptable parameter sets for robust functional operation of this model is the focus of this work. Due to the computational resources and time requirements involved (approximately 30 seconds to 4 minutes per evaluation), a brute force exhaustive search is unfeasible. Using granularity of 1 for parameters with an upper bound greater than 100, 0.1 for an upper bound greater than 2 but less than 100, 0.01 otherwise, and the optimistic estimate of 30 seconds for each evaluation on an eight core computer yields an expected 1.87×10^{20} years to evaluate. We set the goal to investigate the use of efficient stochastic optimisation algorithms. Differential Evolution (DE) [30] is particularly suited to our application. DE is a simple and efficient stochastic vector-based real-value parameter optimisation algorithm with performance (at least) comparable to other leading optimisation algorithms [31], [32]. DE has only two user-defined rates [30], [33], [34], however their optimal values are problem specific and can drastically affect algorithmic performance [35]. This has prompted research into Self-Adaptation (SA), which allows the rates to vary autonomously in a context-sensitive manner throughout an optimisation run. Self-Adaptive DE (SADE) has been shown to perform at least as well as DE on benchmarking problems [35], [36]. Importantly, SA has been shown to reduce the number of evaluations required per optimisation in resource-constrained scenarios with protracted evaluation times [37], compared to non-adaptive solutions. Additionally, we implemented Bayesian optimisation (BO) which models the optimisation space as a Gaussian process and uses a utility function to determine which points to select for optimisation [38]. We compare these optimisers to a uniform random search as a baseline as it has been shown to outperform grid search in some problems [39].

Spiking networks are particularly amenable to a form of unsupervised learning called Spike-Timing Dependent Plasticity (STDP) [40], which allows synaptic weights to change autonomously in response to environmental inputs. STDP has been shown to provide faster responses compared to non-plastic networks in dynamic environments [41], which motivates our investigations into its use in our LGMD networks.

The combination of Evolutionary optimisation with learning, e.g., the Baldwin Effect, is known to be beneficial in artificial and natural systems [42]. In our case off-line optimization (DE or BO) sets up network parametrizations for STDP (learning) to exploit on-line.

Specific use of STDP with meta-optimization is a promising and entirely new area for LGMD networks, which motivates our work in this area. Our hypothesis is that these adaptivity mechanisms are beneficial to the performance of the LGMD network. To test this hypothesis, we evaluate the performance of our optimizers (DE, SADE, and BO, with and without STDP) when optimising looming responses in LGMD networks which are stimulated by (i) simple and (ii) complex DVS recordings on the UAV. Our finding is, however, that the STDP can be both beneficial and destructive for performance of the network.

The original contributions of this work are (i) the development of an objective function that accurately describes the desired LGMD behaviour, (ii) a comprehensive statistical comparisons of three leading algorithms in optimising LGMD networks, and (iii) the first optimisation-based study on the effect of STDP and spike-frequency adaptation in LGMD networks.

II. THE MODEL

This section describes the background for the model set-up, and the specific equations that were used in the experiment.

A. LGMD

We implement the model as described by Salt et al. [26]. The LGMD model consists of a photoreceptor layer (P), a summing layer (S), an intermediate photoreceptor layer (IP), an intermediate summing layer (IS), and an LGMD neuron layer. The intermediate layers can be seen as analogous to sum-pooling layers in deep convolutional neural networks [43]–[45]. These layers are modelled as populations of AEIF neurons, connected by excitatory (E), slow inhibitory (SI), and fast inhibitory (FI) connections. Fig. 1 shows the topology of the network.

The P to IP to LGMD connection inhibits the spiking response of the LGMD to translational motion across the field of vision, and the inhibitory connections (SI and FI) from the photoreceptor to the summing layer inhibit the output neuron from spiking in response to non-looming stimuli. The weights of the inhibitory connections are assigned based on their distance from the central excitatory neuron similarly to that described in [16]. This connection configuration spans the P layer like a kernel.

The intermediate layers were added to make the model compatible with the dynap-se neuromorphic processor described in [4], [6]. However, Salt et al. [26] found that the addition of the intermediate (sum-pooling) layer before the LGMD neuron also increased the performance of the network on all but slow circular stimuli.

1) Adaptive Exponential Integrate and Fire Spiking Networks: We use Adaptive Exponential Integrate and Fire (AEIF) model neurons in the network; the respective neuron equations follow (1) and (2):

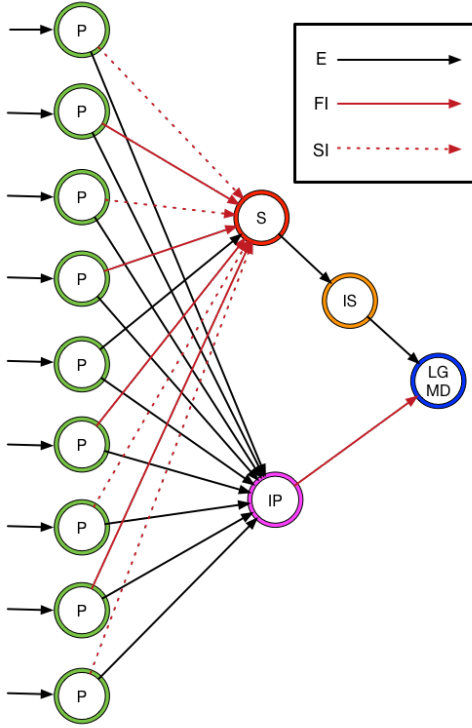$$\frac{dV}{dt} = \frac{-g_L(V - E_L) + g_L \Delta_T \exp(\frac{V - V_T}{\Delta_T}) + I}{C}, \quad (1)$$

Fig. 1: The neuromorphic LGMD model, which consists of a photoreceptor layer (P), a summing layer (S), an intermediate photoreceptor layer (IP), an intermediate summing layer (IS), and an LGMD neuron layer. Edges show connections that are either Excitatory (E), Slow Inhibitory (SI), or Fast Inhibitory (FI).

$$I = I_e - I_{iA} - I_{iB} - I_{adapt}, \tag{2}$$

where C is the membrane capacitance, g_L is the leak conductance, E_L is the leak reversal potential, V_T is the spike threshold, Δ_T is the slope factor, V is the membrane potential, I_e is an excitatory current, I_{adapt} is the adaptation current, and I_{iA} and I_{iB} describe fast and slow inhibitory current, respectively [28]. When a spike is detected ($V > V_T$) the voltage resets ($V = V_r$), and the post-synaptic neuron receives a current injection from the pre-neuron firing given by:

$$I_{e/i,l} = I_{e/i,l} + q_{e/i,l}, \tag{3}$$

$$I_{adapt} = I_{adapt} + b, \tag{4}$$

where the subscript l corresponds to the post-synaptic layer, $q_{e/i,l}$ is the injected current, b is the spike-triggered contribution to adaptation, and the subscript e/i refers to either excitation or inhibition. To simplify the model for embedded implementation, inhibitory currents were set as a ratio of the excitatory current:

$$q_{i,l} = inh_l \cdot q_{e,l}, \tag{5}$$

where inh_l is a constant parameter. This equation holds for both types of inhibition (slow and fast). The decay of the excitatory or inhibitory currents is described by:

$$\frac{dI_{e/i}}{dt} = -\frac{I_{e/i}}{\tau_{e/i}}, \tag{6}$$

where $I_{e/i}$ is the current and $\tau_{e/i}$ is the time constant for the decay. The subscript e/i refers to either excitation or inhibition respectively. Finally, the decay of the adaptation current is described by:

$$\frac{dI_{adapt}}{dt} = \frac{a(V - E_L) - I_{adapt}}{\tau_{adapt}}, \tag{7}$$

where a is the sub-threshold adaptation and τ_{adapt} is the time constant for the decay.

Section IV-B explains how these were implemented and the bounds of all of the values.

B. Spike Timing Dependent Plasticity

Spike Timing Dependent Plasticity (STDP) is a realisation of Hebbian learning based on the temporal correlations between pre- and post-synaptic spikes. This synaptic plasticity is thought to be fundamental to adaptation, learning, and information storage in the brain [46], [47].

Arrival of a pre-synaptic spike closely before a post-synaptic spike increases the efficacy of the synapse, while if a post-synaptic spike is received in close proximity to and before a pre-synaptic spike, the efficacy of the synapse is decreased. Long term potentiating (LTP, synaptic weight increase) of the synapse occurs in the former case, long term depression (LTD, synaptic weight decrease) occurs in the latter case. Fig. 2 shows a typical dependence of the synaptic weight change on the difference in arrival times of the post- and pre- synaptic spikes.
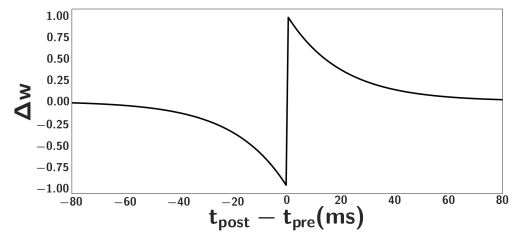

Fig. 2: The impact of STDP on the synaptic weights. If the pre-synaptic spike arrives before the post synaptic spike, then the strength of the weights is increased. If the post synaptic spike arrives first, then the strength of the synapse is weakened.

STDP modifies the synaptic current injection given in (3) by multiplying it by a weight w, which is modified according to the plasticity rule. In particular, if a pre-synaptic spike occurs, then:

$$I_{e/i,l} = I_{e/i,l} + wq_{e/i,l}, \tag{8}$$

$$A_{pre} = A_{pre} + \Delta_{pre}, \tag{9}$$

$$w = w + A_{pre}. \tag{10}$$

If a post-synaptic spike occurs then:

$$A_{post} = A_{post} + \Delta_{post}, \qquad (11)$$

$$w = w - A_{post}. \qquad (12)$$

$A_{pre|post}$ are the amount by which the weight w is strengthened or weakened, and $\Delta_{pre|post}$ is a user-defined value for increasing $A_{pre|post}$ each time a spike occurs. At each spike event, the variables $A_{pre|post}$ decay:

$$\frac{dA_{pre|post}}{dt} = -\frac{A_{pre|post}}{\tau_{pre|post}}. \qquad (13)$$

This learning rule leads to potentiation of synapses that are supported by temporal sequence of pre- and post-synaptic spikes and depression of synapses that connect neurons that fire in a reverse order. In other words, the connection strengths vary depending on the activity of the neurons they are connected to.

III. OPTIMISATION TECHNIQUES

In this Section, we describe the three optimisation techniques that we compare: DE, SADE, and BO, and how they are applied to optimising the LGMDNN parameter space. Each individual, x_i, is a parametrisation of the LGMDNN, given by: $x_i = [\tau_{\mathbf{e}}, \tau_{\mathbf{iA}}, \tau_{\mathbf{iB}}, \mathbf{q_{eP}}, \mathbf{q_{eS}}, \mathbf{q_{eIP}}, \mathbf{q_{eIS}}, \mathbf{q_{eL}}, \mathbf{inhA_S},$ $\mathbf{inhB_S}, \mathbf{inhA_L}, [\mathbf{a}, \mathbf{b}, \tau_{\mathbf{w_{adapt}}}], (\tau_{\mathbf{pre}}, \tau_{\mathbf{post}}, \mathbf{\Delta_{pre}}, \mathbf{\Delta_{post}})]$. The bounds for each element of x_i can be found in Table I in Section IV-B, as well as a brief explanation of the meaning of each parameter.

A. Differential Evolution

DE is an efficient and high performing optimiser for real-valued parameters [30], [33]. As it is based on evolutionary computing, it performs well on multi-modal, discontinuous optimisation landscapes. Storn and Price [30] showed that their original DE outperformed several other stochastic optimization techniques in benchmarking tests whilst requiring the setting of only two parameters, crossover probability CR and differential weight F. It also requires a mutation function, which determines how individuals in the population are mixed. Many variants of the mutation function have been suggested, these follow the naming convention $DE/x/y/z$. Here we use DE rand/1/bin where x denotes the vector to be mutated (in this case a random vector), y denotes the number of vectors used, and z denotes the crossover method (bin corresponds to binomial).

The initial population, $\mathbf{X_1} = \{\mathbf{x}_{1,1}, \mathbf{x}_{2,1}, ..., \mathbf{x}_{NP,1}\}$, where NP is the size of the population and $\mathbf{x}_{i,1} \in \mathcal{R}^D$ is an individual that contains the D parameters to be optimised, is generated from random samples drawn from a uniform probability distribution of the parameter space, bounded to the range of the respective variable. These bounds are shown in Section IV-B3. The fitness of each vector in the population is calculated by the objective function, as described in Section IV-A.

In each generation, each parent generates one offspring by way of a 'donor' vector, created following Eq. (14):

$$v_{i,G+1} = x_{r_1,G} + F \cdot (x_{r_2,G} - x_{r_3,G}), \qquad (14)$$

where $r_1 \neq r_2 \neq r_3 \neq i \in [1, NP]$ index random unique population members, the subscript G indicates the current generation, and differential weight $F \in [0, 2]$ determines the magnitude of the mutation. The final offspring is generated by probabilistically merging elements of the parent with elements of the donor vector. The new vector $u_{i,G+1} = (u_{1i,G+1}, \ldots, u_{Di,G+1})$ is found by:

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1}, & \text{if } rand(j) \leq CR \text{ or } j = R, \\ x_{ji,G}, & \text{otherwise}, \end{cases} \qquad (15)$$

where $j \in (1, \ldots, D)$, $CR \in [0, 1]$ is the crossover rate, $rand(j) \in [0, 1]$ is a uniform random number generator, and $R \in (1, \ldots, D)$ is a randomly chosen index to ensure that at least one parameter changes. The value of offspring with index i is then calculated as:

$$x_{i,G+1} = \begin{cases} u_{i,G+1}, & \text{if } f(u_{ji,G+1}) > f(x_{i,G}), \\ x_{i,G}, & \text{otherwise}, \end{cases} \qquad (16)$$

where $f(\cdot)$ is the fitness function. Once all offspring are generated, they are evaluated on the fitness function, and selected into the next generation if they score better than their parent. Otherwise, the parent remains in the population.

B. Self-Adaptive DE

Brest et al. [35] present the first widely-used self-adaptive rate-varying DE, which is expanded by Qin et al., to allow the mutation scheme to be selected (from four predetermined schemes) alongside the rates [36], based on previously successful settings. Different rates/schemes are shown to work better on different problems, or in different stages of a single optimisation run. The strategy for a given candidate is selected based on a probability distribution determined by the success rate of a given strategy over a learning period LP. A strategy is considered successful when it improves the individual's value. In the interest of brevity, we refer the interested reader to [36] for a full algorithmic description.

Rates are adapted as follows. Before $G > LP$ (where G is number of generations, and LP is the number of generations needed before the learned CR values are used), CR is calculated by randomly selecting a number from a normal distribution, $N(0.5, 0.3)$, with a mean of 0.5 and a standard deviation of 0.3. Afterwards it is calculated by a random number from $N(CR_{mk}, 0.1)$ where CR_{mk} is the median value of the successful CR values for each strategy K. F is simply selected from a normal distribution $N(0.5, 0.3)$, which will cause it fall on the interval $[-0.4, 1.4]$ with a probability of 0.997 [36].

C. Bayesian Optimisation

Bayesian optimisation (BO), e.g. [38], is a probabilistic optimisation process that typically requires relatively few

evaluations [48]–[50], although the evaluations themselves are computationally expensive. When parallelised, BO is shown to locate hyper-parameters within set error bounds significantly faster than other state-of-the-art methods on four challenging ML problems [51], in one case displaying 3% improved performance over state-of-the-art expert results. As such, BO can be considered a competitive optimiser to which we can compare DE and SADE.

BO assumes the network hyper-parameters are sampled from a Gaussian process (GP), and updates a prior distribution of the parameterisation based on observations. For LGMDNN, observations are the measure of generalization performance under different settings of the hyper-parameters we wish to optimise. BO exploits the prior model to decide the next set of hyper-parameters to sample.

BO comprises three parts: (i) a prior distribution, (ii) an acquisition function, and (iii) a covariance function.

1) Prior: We use a Gaussian Process (GP) prior, as it is particularly suited to optimisation tasks [48]. A GP is a distribution over functions specified by its mean, m, and covariance, k, which are updated as hyper-parameter sets are evaluated. The GP returns m and k in place of the standard function f:

$$f(x) \sim GP(m(x), k(x, x')). \quad (17)$$

2) Covariance Function: The covariance function determines the distribution of samples drawn from the GP [38], [51]. Following [51], we select the 5/2 ARD Matérn kernel (18), where θ is the covariance amplitude.

$$k_{m52}(x_i, x_j) = \eta exp(-\sqrt{5r^2(x_i, x_j)}), \quad (18)$$

where:

$$\eta = \theta(1 + \sqrt{5r^2(x_i, x_j)} + \frac{5}{3}r^2(x_i, x_j)), \quad (19)$$

where:

$$r^2(x_i, x_j) = \frac{x_i - x_j}{\theta^2}. \quad (20)$$

3) Acquisition Function: An acquisition function is a function that selects which point in the optimisation space to evaluate next. We evaluate the three acquisition functions, which select the hyper-parameters for the next experiment: Probability of Improvement (PI), Expected Improvement (EI) [48], and Upper Confidence Bound (UCB) [52] — see [38] for full implementation details. $\mu(\cdot)$ and $\sigma(\cdot)$ refer to the mean and standard deviation functions.

Briefly, the **PI** can be calculated, given our current maximum observation of the GP, x^+, by:

$$\begin{aligned} PI(x) &= P(f(x) \geq f(x^+) + \zeta) \\ &= \Phi(\frac{\mu(x) - f(x^+) - \zeta}{\sigma(x)}). \end{aligned} \quad (21)$$

Where, $\Phi(.)$ is the normal cumulative distribution function. Here, $\zeta \geq 0$ is a user-defined trade-off parameter that balances exploration and exploitation [53].

Similarly, **EI** is evaluated by [54]:

$$EI(x) = \begin{cases} ei + \sigma(x)\phi(Z), & \text{if } \sigma(x) > 0, \\ 0, & \text{otherwise}; \end{cases} \quad (22)$$

$$ei = (\mu(x) - f(x^+) - \zeta)\Phi(Z); \quad (23)$$

$$Z = \begin{cases} \frac{\mu - f(x^+) - \zeta}{\sigma(x)}, & \text{if } \sigma(x) > 0, \\ 0, & \text{otherwise}, \end{cases}$$

where ϕ and Φ correspond to the probability and cumulative distribution functions of the normal distribution, respectively.

UCB maximises the upper confidence bound:

$$UCB(x) = \mu(x) + \kappa\sigma(x), \quad (24)$$

where $\kappa \geq 0$ balances exploration and exploitation [51], and is calculated per evaluation as:

$$\kappa = \sqrt{\nu\tau_t}, \quad (25)$$

where ν is the user tunable variable and:

$$\tau_t = 2\log(\frac{t^{\frac{d}{2}+2}\pi^2}{3\delta}). \quad (26)$$

$\delta \in \{0, 1\}$, d is the number of dimensions in the function and t is the iteration number.

IV. TEST PROBLEM

This section outlines the rationale of the objective function, the experimental set-up, and assumptions. It is important to note that the motivation behind the model simplifications and objective function is for the work to be directly transferable to the neuromorphic processors described in [29] once they are readily available.

A. Objective Function

Initially the optimisation function was formulated as:

$$F_{init}(\lambda) = Acc - ||V||_2, \quad (27)$$

where Acc is the accuracy given by Eq. (31), $||V||_2$ is the regularisation term, in particular the l2-norm of the voltage signal, used to regularise the voltage signal, and λ is a candidate solution. However, this objective function resulted in all optimisers producing 50% accuracy with looming detected at any time in the experiments.

To improve the accuracy, the function to optimise was formulated as a weighted multi-objective function [55]. The objective function has three distinct parts: accuracy (Eq. (31)), sum squared error of the membrane voltage signal (Eq. (36)), and the reward of the spiking trace (Eq. (34)). The accuracy alone could not be used as there were only eight discrete events in the input stimuli during the optimisation phase which was not granular enough for optimisation. Eq. (36) acts as a regularising term to prevent the voltage trace from becoming too large. Eq. (34) is used to rate the spiking behaviour with more granularity than is possible with accuracy. Combining Eq. (36) and (34) resulted in spiking behaviour with a realistic voltage trace. The accuracy was then used to account for sub-optimal regions of (34) that still resulted in a high score.

This resulted in the final formulation of the objective function modified by accuracy, $F_{Acc}(\lambda)$, which is calculated by:

$$F_{Acc}(\lambda) = \begin{cases} 2 \cdot F(\lambda), & \text{if } F(\lambda) > 0 \text{ and } Acc = 1, \\ Acc \cdot F(\lambda), & \text{if } F(\lambda) < 0, \\ 0, & \text{if } Acc = 1 \text{ and } F(\lambda) < 0, \\ F(\lambda), & \text{otherwise.} \end{cases} \tag{28}$$

Here, Acc is the accuracy of the LGMDNN output and $F(\lambda)$ is the fitness function. The LGMD network is said to have detected a looming stimulus if the output neuron's spike rate exceeds a threshold SL. This can be formalised by:

$$Looming = \begin{cases} \text{True}, & \text{if } SR > SL, \\ \text{False}, & \text{otherwise,} \end{cases} \tag{29}$$

where SR can be calculated by:

$$SR = \sum_{i=t}^{t+\Delta T} S_i, \tag{30}$$

where ΔT is the time over which the rate is calculated and S_i is whether or not there is a spike at time i; a spike is defined to occur if at time i the membrane potential exceeds V_T (V_T has the same meaning as in Eq. (1)).

The looming outputs are categorised into true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). Output accuracy is then:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}. \tag{31}$$

The fitness function without accuracy, $F(\lambda)$, can be calculated by:

$$F(\lambda) = \frac{Score(\lambda) + SSEOS(\lambda)}{2}, \tag{32}$$

where $Score$ is a scoring function based on the timing of spiking outputs and $SSEOS$ is the sum squared error of the output signal.

The score is calculated by difference of the penalties' and reward functions' sums over the simulation:

$$Score(\lambda) = \sum_{i=1}^{N} R_i - \sum_{i=1}^{N} P_i. \tag{33}$$

The reward at a given time can be calculated by:

$$R(t) = \begin{cases} k \exp(\frac{t}{\Delta t}) + 1, & \text{if looming and spike,} \\ 0, & \text{otherwise.} \end{cases} \tag{34}$$

The punishment can be calculated by:

$$P(t) = \begin{cases} (l - c)\frac{t}{\Delta t} + c, & \text{if not looming and} \\ & \text{spike and } t < \frac{\Delta t}{2}; \\ (l - c)\frac{1-(t-\frac{\Delta t}{2})}{\frac{\Delta t}{2}} + c, & \text{if not looming} \\ & \text{and spike and } t > \frac{\Delta t}{2}; \\ 0, & \text{otherwise.} \end{cases} \tag{35}$$

In these equations, t and Δt remain consistent with the other objective functions and k, l, and c are all adjustable constants

to change the level of punishment or reward.

To calculate $SSEOS(\lambda)$, the signal was first processed so that every spike had the same value. This was done so that the ideal voltage and the actual voltage would match in looming regions, as the voltage can vary for a given spike. Ultimately, the only criterion is that the voltage has crossed the spiking threshold. In the non-looming region the ideal signal was taken to be the resting potential, which was negative for the AEIF model equation. The signal error was calculated at every time step as:

$$SSEOS(\lambda) = -\sum_{i=1}^{N}(V_{actual}^{i} - V_{ideal}^{i})^2. \tag{36}$$

V_{actual} could be obtained directly from the state monitor object of the LGMD output neuron in the SNN simulator (Brian2). N in this case is the length of the simulation and i indicates each recorded data point at each time step of the simulation. V_{ideal} was given by:

$$V_{ideal} = \begin{cases} V_{spk}, & \text{if looming,} \\ V_r, & \text{otherwise,} \end{cases} \tag{37}$$

where V_{spk} is the normalised value given to each spike and V_r is the resting potential.

Overall, this gives an objective function that takes into account the expected spiking behaviour, whilst penalising the system for deviating from plausible voltage values and rewarding it for accurately categorising looming and non-looming stimuli. The voltage signal was kept to realistic bounds as the model was designed to target neuromorphic processors such as the ROLLS chip [4].

B. Experimental Set-up

The model was set-up using the Python Brian2 spiking neural network simulator [56].

1) Data Collection: Data was collected using a DVS in-situ mounted on a quadrotor UAV (QUAV). Two types of data were collected: simple and real world. The simple data was synthesised using PyGame to generate black shapes on a white background that increased in area in the field of view of the DVS. This included: a fast and slow circle, a fast and slow square, and a circle that loomed then translated while increasing in speed (composite). The laptop playing the stimuli was placed in front of the hovering QUAV and the stimuli were recorded. This was done to maintain any noise that might be generated by the propellers of the QUAV.

To challenge the model, real stimuli were also recorded: a white ball on a black slope was rolled towards the DVS from 3 different directions; a cup was suspended in the air in front of the hovering QUAV and moving towards and away from the QUAV; and a hand was moved towards and away from the DVS on the hovering QUAV. These are increasing in complexity in terms of the shapes that are presented.

Four looming and non-looming events ($\sim 25\,$s) from the composite stimulus were used to optimise the model and then the optimised model was evaluated on the other stimuli. The stimuli were chosen to show that the model generated is both shape and speed invariant.

2) Experimental Constants: ΔT from Eq. (30) was set to be 10 ms. A loom was said to be detected if SR from Eq. (30) exceeded 13. This had to occur before the last 10% of the looming sequence to allow enough time for the UAV to react. The clock in Brian was set to have 0.1 ms granularity. This meant that the model could react after 1.3 ms if the loom was intense, at most it would take 10 ms.

3) Hyper-parameter constraints: The hyper-parameters were all continuous and could range from zero to infinity. There were many regions of the parameter space that were not computable even when using a cluster with 368GB of RAM. To mitigate some of the computational difficulties, Bayesian optimisation using the expected improvement utility function (BO-EI) was used over 20 eight hour runs to find feasible regions of the hyper-parameter space allowing us to constrain the optimisation space.

C, g_L, E_L, V_T, and Δ_T are parameters of the neuron equation, not the model, and were set as constants; performance was not impacted by setting these values and appropriately optimising the other parameters [57]: $C = 124.2\,pF$, $g_L = 60.05\,nS$, $E_L = -73.12\,mV$, $V_T = -3.98\,mV$, and $\Delta_T = 6.71\,mV$.

Table I shows the constraints found for the rest of the hyper-parameters.

TABLE I: Parameters of the optimisation space and their constraints.

Param.	Min	Max	Description
$\tau_{\mathbf{e}}$	1	10	Decay time const, exc.
$\tau_{\mathbf{iA}}$	1	20	Decay time const, A inh.
$\tau_{\mathbf{iB}}$	1	25	Decay time const, B inh.
$\mathbf{q_{eP}}$	0	1363	Exc. current inj. to P
$\mathbf{q_{eS}}$	0	5000	Exc. current inj. to S
$\mathbf{q_{eIP}}$	0	230	Exc. current inj. to IP
$\mathbf{q_{eIS}}$	0	270	Exc. current inj. to IS
$\mathbf{q_{eL}}$	0	472	Exc. current inj. to L
$\mathbf{inhA_S}$	0.04	1.22	Inh.A/Exc. ratio for S
$\mathbf{inhB_S}$	0.24	1.5	Inh.B/Exc. ratio for S
$\mathbf{inhA_L}$	0.019	1.3	Inh.A/Exc. ratio for L
$\mathbf{a}$	1	8	Sub-thresh. adaptation
$\mathbf{b}$	40	141	Spike-contrib. to adapt.
$\tau_{\mathbf{w_{adapt}}}$	1	150	Time const, adapt.
$\tau_{\mathbf{pre}}$	1	25	Time const of A_{pre}
$\tau_{\mathbf{post}}$	1	25	Time const of A_{post}
$\mathbf{\Delta_{pre}}$	1e-16	0.05	Current+ at pre-syn. spike
$\mathbf{\Delta_{post}}$	1e-16	0.05	Current+ at post-syn. spike

4) Comparing optimisers: SADE, DE, BO with EI(BEI), BO with POI (BPOI), and BO with UCB (BUCB) were evaluated thirty times on the same input stimulus, so that they could be statistically compared using a Mann-Whitney U test. A random search (RNG) was also ran in the same manner as a benchmark for the algorithms, which has been shown to be a natural baseline with which to compare other optimisation algorithms as it can outperform grid searches in terms of results and number of calculations [39]. The input stimulus included a black circle on a white background performing a

eight looming and eight non-looming events. The non-looming events contained a combination of a shrinking circle and a translating circle from left to right or right to left. The values of the user defined parameters were selected as:

- BEI and BPOI: $\zeta = 0.01$;
- BUCB: $\kappa = 2.576$;
- DE: $NP = \frac{10dim}{3}$, $F = 0.6607$, $CR = 0.9426$;
- SADE: $LP = 3$, $NP = \frac{10dim}{3}$, where dim is the number of hyper parameters;
- RNG: Individuals were selected from a uniform distribution.

Note, that we have chosen all user-defined parameters based on values previously used in the literature.

The tests were run using the non-adaptive and non-plastic model with the bounds from Table I. They were defined as having converged if they had not improved for $10 \times NP$ evaluations. This meant ten generations for the DE algorithms and the same number of BO or RNG evaluations evaluations. The population size was two more than what is recommended by [34] for the DE algorithm. This size was chosen as it is relatively small and time was an issue. The short convergence meant that the SADE algorithm needed to have a short LP. The processor time was not included as a metric for this as the tests were run on three different computers so the results would not have been comparable.

5) Comparing Models: Once the best optimiser was found (a comparison of optimisers can be found in Subsection V-A), the best performing optimiser, SADE, was used to optimise the following models:

LGMD: Neuromorphic LGMD;
A: LGMD with adaptation;
P: LGMD with plasticity;
AP: LGMD with adaptation and plasticity.

The SADE variables were set to: $LP = 3$ and $NP = 10$. The optimisation process was run 10 times and the best optimiser from these ten runs was selected. The model was then tested on each input case for ten looming to non-looming or non-looming to looming transitions. The performance of each model is reported in Subsection V-C.

Plasticity was found to degrade the performance sometimes so we experimented clamping it from 0% to 100% of the original synaptic strength. This allowed the weights to range from zero to double the original values when at 100% STDP to no variation at 0% STDP.

V. RESULTS AND DISCUSSION

The results are split into two subsections. First, we will compare the optimisers and then we will compare the addition of adaptation, plasticity, and adaptation and plasticity combined to the baseline model.

The models are evaluated on their accuracy (Acc), sensitivity (Sen), Precision (Pre), and Specificity (Spe). Acc is defined in Subsection IV-A. The other metrics can be found in [58].

A. Optimiser comparison and statistical analysis

Table II shows that the SADE algorithm achieved the best fitness. A good fitness value is one that is greater than 0, this

means that it is either 100% accurate or exhibits a desirable voltage trace The optimisers all achieved a negative fitness values due to decreased population size and convergence conditions to enable us to run enough experiments to be able to perform statistical analysis. Section V-C shows the results of SADE when run with a less restrictive convergence condition. The DE algorithm converged on the worst solution in the least number of objective function evaluations, and achieved the best specificity but the worst fitness, precision, and sensitivity. RNG achieved the highest average accuracy, sensitivity, and precision. BPOI and SADE still had greater fitness. As the fitness function is negative it is likely that it was rarely modified by the accuracy as per Eq. (28). All of the models were able to find locations in the optimisation space with 100% accuracy/sensitivity given enough time, however as evaluations are time-expensive, not all these algorithms may be viable and a combination of performance metrics should be considered.

TABLE II: Optimisation algorithm metrics.

Meth	Fit	Eva	Acc	Sen	Pre	Spe
DE	-3779.96	**440.40**	0.52	0.06	0.18	**0.98**
BEI	-2653.45	761.29	0.65	0.39	0.70	0.90
SADE	**-1749.83**	1345.66	0.61	0.33	0.58	0.89
RNG	-2328.15	754.80	**0.67**	**0.47**	**0.75**	0.86
BUCB	-2433.38	832.40	0.60	0.43	0.63	0.77
BPOI	-1968.19	794.27	0.63	0.40	0.71	0.86

Table III shows the statistical significance of the results from Table II. The method in the comparison column is compared to each method in the subsequent column. A + indicates statistically significant values and a . indicates no statistical significance. Statistical significance was defined as $p \leq 0.05$. The Mann-Whitney U test was used to determine statistical significance because it does not require normally distributed samples.

SADE's fitness was significantly better compared to all optimisers, but it also performed the most evaluations. This difference is statistically significant. SADE had the second best accuracy but this is only significantly different to RNG and DE. Its sensitivity was significantly worse than BUCB and RNG. It had significantly worse precision than RNG, but significantly better specificity.

BPOI was significantly better than DE, BEI, and BUCB for fitness. It also had significantly less evaluations than SADE.

BEI had significantly worse fitness and sensitivity when compared to BOPI, RNG and SADE. Interestingly, even though it had the second best accuracy the difference is only significant between it and BUCB and SADE.

DE took significantly fewer evaluations to converge when compared to all algorithms but also had significantly worse fitness, accuracy, sensitivity, and precision. However it had significantly better specificity than all other algorithms. DE was statistically different to every algorithm for every metric.

BUCB had slightly worse fitness than RNG but it is not statistically significant. Its accuracy was significantly worse than BEI but significantly better than DE.

TABLE III: Comparison of the statistical significance of the results.

Meth		Fit	Eva	Acc	Sen	Pre	Spe
BUCB	DE	+	+	+	+	+	+
	BEI	.	.	+	.	.	+
	SADE	+	+	.	+	.	+
	RNG	.	+	+	.	+	+
	BPOI	+	.	.	.	.	+
DE	BEI	+	+	+	+	+	+
	SADE	+	+	+	+	+	+
	RNG	+	+	+	+	+	+
	BUCB	+	+	+	+	+	+
	BPOI	+	+	+	+	+	+
BEI	DE	+	+	+	+	+	+
	SADE	+	+	.	.	.	.
	RNG	+	.	.	+	.	.
	BUCB	.	.	+	.	.	+
	BPOI	+	.	.	.	.	.
SADE	DE	+	+	+	+	+	+
	BEI	+	+	.	.	.	.
	RNG	+	+	+	+	+	.
	BUCB	+	+	.	+	.	+
	BPOI	+	+	.	.	.	.
BPOI	DE	+	+	+	+	+	+
	BEI	+	.	.	.	.	.
	SADE	+	+	.	.	.	.
	RNG	.	.	.	+	.	.
	BUCB	+	.	.	.	.	+
RNG	DE	+	+	+	+	+	+
	BEI	+	.	.	+	.	.
	SADE	+	+	+	+	+	.
	BUCB	.	+	+	.	+	+
	BPOI	.	.	.	+	.	.

A possible reason that DE underperformed is that the F values provided in [34] are not appropriate for this problem. The population size may have also been too small, as populations were truncated to one third of the size recommended in [34]. A smaller population was used for a fair comparison to the Bayesian optimisation algorithms which have a higher computational overhead than DE and SADE. SADE outperformed DE with the same sized population and may have performed better given a larger population. Increasing the population size meant that BPOI and BUCB were not able to complete one run in the time it took SADE, DE, and RNG to do thirty. EI was only able to complete 25 of 30 runs with the larger population. The Bayesian optimisers don't have a population size but the stopping condition was based on it.

SADE removes the need to find control parameters and has been shown to perform as well or better than DE even when the control parameters are well selected [36]. The generalisability that comes with finding the right control parameters on-the-fly is also appealing.

The addition of the various mutation functions to SADE also seems to help it find better results. This is likely due to

the desirable properties of each mutation function cancelling out the undesirable properties of other mutation functions.

A surprising result was that of the BO algorithms BPOI seemed to perform the best. This contrasts to previous studies [38] that ranked it last compared to BEI and BUCB. BPOI tends to focus more on exploitation rather than exploration, choosing regions in the GP with a higher mean rather than variance.

Fig. 3 also shows that SADE's CR and F values converged on small values indicating that it also preferred exploitation to exploration. We postulate that the surface has multiple sharp peaks making it difficult for the optimisers to find good values, but finding one of these peaks and climbing it yields better results than being overly exploratory. For example, if BO landed near a sharp peak using UCB or EI it would never be inclined to evaluate nearby points, because the utility function tends to prioritise regions of higher uncertainty and uncertainty may be lower near evaluated points. RNG offers an unbiased search through the space and performs worse, in terms of fitness, than the exploitative algorithms but better than the exploratory ones.

B. SADE Averages

The SADE algorithm performed the best, in terms of fitness, out of all of the algorithms. Fig. 3a shows the average F_{acc} of the population over 19 generations. The average F_{acc} converged by five iterations. The maximum F_{acc} starts off at 0. This indicates that a 100% accuracy candidate was found in the initialisation period. The maximum F_{acc} then rises to 400 which is not visible as the range of the average score is -50000 to -1500.

The F average results in Fig. 3b are quite interesting. They start off at 1 as they are selected from $U([0,2])$ and then drop down to 0.5 as they are selected from $U([0,1])$ after the first generation. Once the learning period has finished all of the F values have converged to less than 0.1. This indicates that the F values that are having the most success are small and therefore taking advantage of exploration rather than exploitation. It was unexpected that the algorithm would find a min/max within so few generations. This could be why the authors select initial F from $N(0.5, 0.3)$ with range [-0.4,1.4].

Fig. 3c shows how the crossover probability CR for each function changes over time. For the first nine generations, the CR values are selected from $U([0,1])$ and so the mean stays at 0.5. However, as with the F mean values once the learning period is over, all of the CR values go down to less than 0.1. This means that less than 10% of the mutations will generally take place. From a set of 11 hyper-parameters this means that probabilistically one value will change in addition to the random index that is chosen.

The probability of each function being chosen is shown in Fig. 3d. The probabilities are fixed at 0.25 for the first 9 generations and then they vary based on their success. It is interesting to see that in spite of the F and CR values suggesting that the algorithm is converging on a solution, the DE/Rand-to-Best/2/Bin algorithm is the least successful. The DE/Curr-to-Rand/1 algorithm performs relatively well until about 12
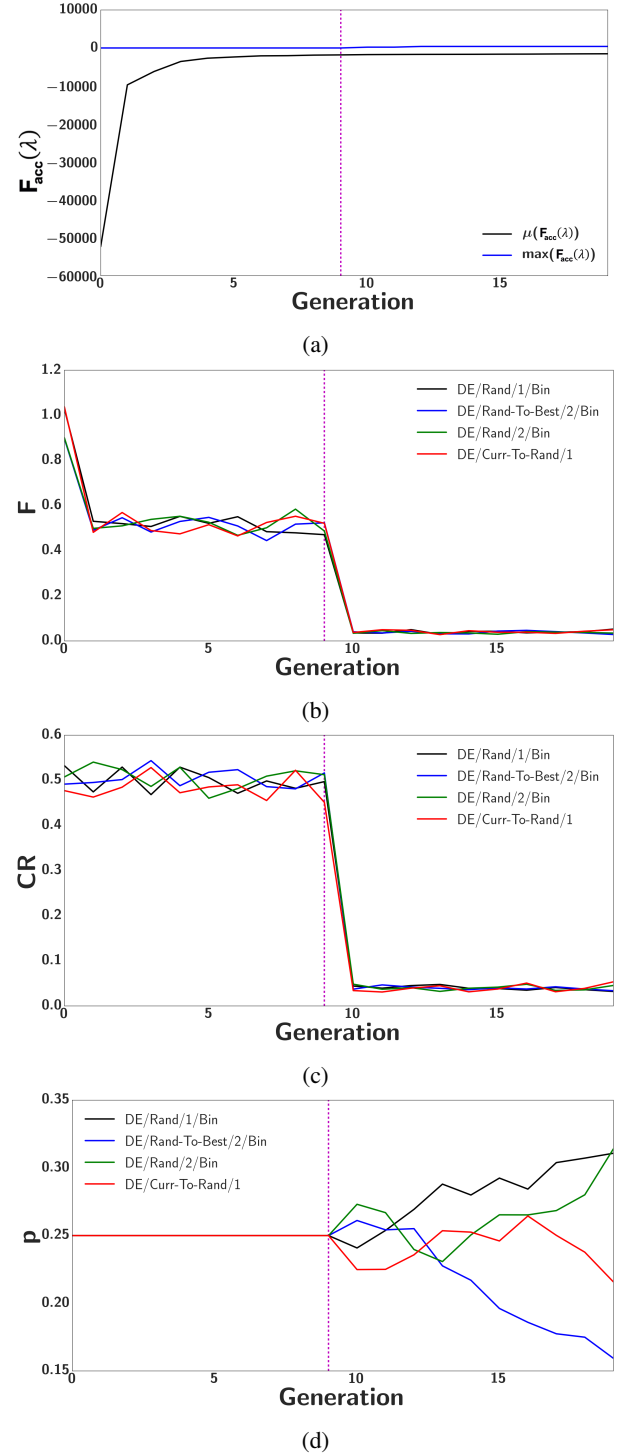

(a)

(b)

(c)

(d)

Fig. 3: Averages $F_{acc}(\lambda)$, F, CR, and p for the SADE population over 19 generations. The dotted vertical line indicates that the learning period has ended. Note description of the SADE algorithm in Section III-B.

TABLE IV: Parameters used by each model.

Parameter	LGMD	A	P	AP
$\tau_e(\mathbf{ms})$	5.87	5.87	5.87	5.87
$\tau_{iA}(\mathbf{ms})$	3.57	3.57	3.57	3.57
$\tau_{iB}(\mathbf{ms})$	4.20	4.20	4.20	4.20
$q_{eP}(\mathbf{pA})$	1014.00	1014.00	1014.00	1014.00
$q_{eS}(\mathbf{pA})$	4635.30	4635.30	4635.30	4635.30
$q_{eIP}(\mathbf{pA})$	84.26	84.26	84.26	84.26
$q_{eIS}(\mathbf{pA})$	168.11	168.11	168.11	168.11
$q_{eL}(\mathbf{pA})$	80.00	100.00	80.00	100.00
$inhA_S(\mathbf{1})$	1.19	1.19	1.19	1.19
$inhB_S(\mathbf{1})$	1.50	1.50	1.50	1.50
$inhA_L(\mathbf{1})$	0.14	0.14	0.14	0.14
$a(\mathbf{1})$	-	0.79	-	0.79
$b(\mathbf{1})$	-	14.51	-	14.51
$\tau_{w_{adapt}}(\mathbf{ms})$	-	30.00	-	30.00
$\tau_{pre}(\mathbf{ms})$	-	-	1.56	1.56
$\tau_{post}(\mathbf{ms})$	-	-	10.03	10.03
$\Delta_{pre}(\mathbf{1})$	-	-	0.031	0.031
$\Delta_{post}(\mathbf{1})$	-	-	0.027	0.027
$c(\mathbf{1})$	-	-	0.05	0.05

TABLE V: Quality metrics of the performance of different LGMD models for different simulated looming stimuli.

Stimulus	Model	Acc	Sen	Pre	Spe
comp	LGMD	0.90	1.00	0.83	0.80
	A	0.90	1.00	0.83	0.80
	P	0.90	1.00	0.83	0.80
	AP	0.90	1.00	0.83	0.80
circleSlow	LGMD	0.80	0.60	1.00	1.00
	A	0.80	0.60	1.00	1.00
	P	0.90	0.80	1.00	1.00
	AP	1.00	1.00	1.00	1.00
circleFast	LGMD	1.00	1.00	1.00	1.00
	A	1.00	1.00	1.00	1.00
	P	1.00	1.00	1.00	1.00
	AP	1.00	1.00	1.00	1.00
squareSlow	LGMD	1.00	1.00	1.00	1.00
	A	1.00	1.00	1.00	1.00
	P	1.00	1.00	1.00	1.00
	AP	1.00	1.00	1.00	1.00
squareFast	LGMD	1.00	1.00	1.00	1.00
	A	1.00	1.00	1.00	1.00
	P	1.00	1.00	1.00	1.00
	AP	1.00	1.00	1.00	1.00

generations where it tapers off. The DE/Rand/2/bin algorithm dips initially but then increases as DE/Curr-to-Rand/1 starts to drop off. The DE/Rand/1/bin remains relatively high during the entire algorithm only to be overtaken by The DE/Rand/2/bin in the last generation.

C. Comparison of models

Table IV shows the selected final parameters of each model. These values were all found by the SADE algorithm, due to the superior quality of its results. The (1) tag in the parameter column indicates that the variable is unit-less.

In both models with plasticity, the clamping value c was set to 0.05, or 5%.

As expected, all of the models have a $\tau_{iA} < \tau_{iB}$ which means that the B inhibitions will persist for longer and have slower dynamics relative to the A inhibitions. What is unexpected is that the B inhibitions also have stronger current injection than the A inhibitions. On top of this, both of the inhibitory current injections are actually stronger than the excitatory connections. Whereas the model in [24] with discrete dynamics had relatively low inhibitory current injections, with $inhA_S = 0.25$ and $inhB_S = 0.125$ of the excitation strength. Clearly, there is a difference between the neuron models that are used, but this is an interesting outcome nonetheless.

Table V shows the accuracy, sensitivity, precision, and specificity for each LGMD model for a given simple stimulus. The stimuli can be described as follows:

composite: A standard test bench stimulus that consists of a black circle on a white background that translates and looms at increasing speeds. Fig. 4a shows the composite input.

circleFast/Slow: A purely looming black circle on white background at high or low speeds. Collected on hovering QUAV. Fig. 4b shows the circleFast/Slow stimulus.

squareFast/Slow: A purely looming black square on a white background at high/low speeds. Fig. 4c shows the square-Fast/Slow stimulus.

The results in Table V show that the models performed well ($Accuracy \geq 0.8$) on most of the stimuli. LGMD and **A** perform poorly on the circleSlow test, missing two out of five of the looming stimuli. **P** misses one looming stimulus, and **AP** detects all stimuli accurately. The plasticity increases the weights of important connections and the adaptation filters out over excited neurons.

These results show that the models are capable of detecting looming stimuli of varying speeds and of differentiating between translation and looming stimuli for the most part. **AP** scored 100% in every test besides the composite stimulus where it misclassified the first short translation as a loom. It is likely that this is due to the network not starting in its resting/equilibrium state. Inspecting the output trace of **LGMD** on the **composite** stimulus it takes the model $\tilde{1}50ms$ to stop spiking after the looming phase has finished.

After performing the simulated experiments of computer generated shapes, real objects moving towards and away from the camera were recorded. These stimuli can be described as:

ballRoll[1-3]: Three different runs of a white ball rolling towards the camera on a black platform at different angles and speeds. This is a purely looming stimulus. Fig. 5a shows one of the three ball rolls.
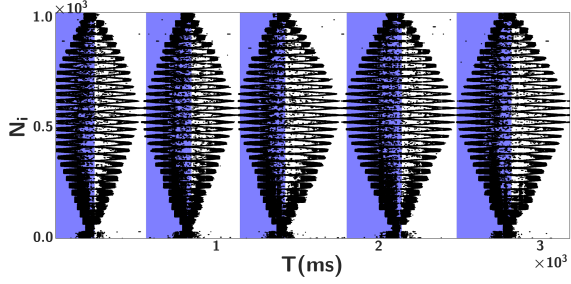
cupQUAV: A QUAV flying towards a cup suspended in front of it with a white wall behind it. This is a self stimulus. Fig. 5b shows the QUAV cup stimulus.
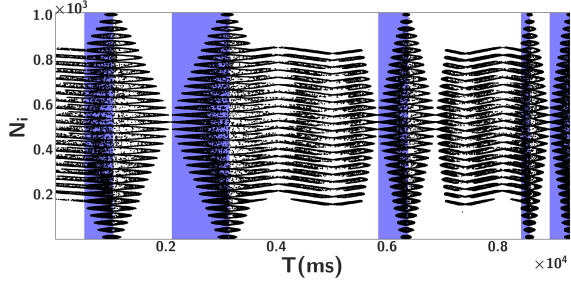
(a) Filtered Composite Input (P Layer Raster Plot).
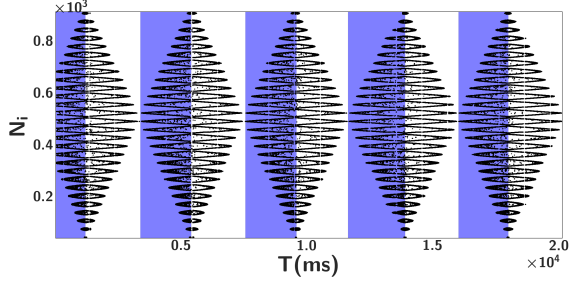
(b) Filtered circleSlow Input (P Layer Raster Plot).

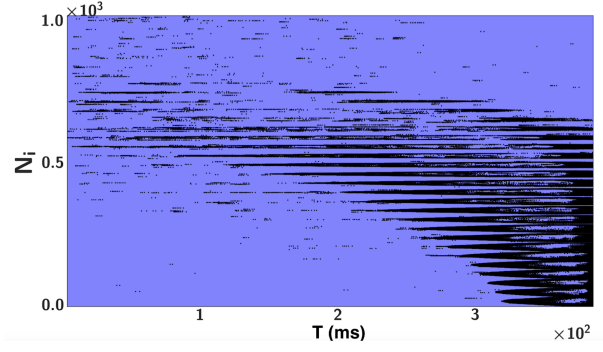
(c) Filtered squareFast Input (P Layer Raster Plot).

Fig. 4: The input layer for the simple stimuli.The white and coloured backgrounds indicate non-looming and looming respectively.

Hand: A Hand moving towards and away from the hovering QUAV. Fig. 5c shows the looming hand stimulus.

Fig. 5a, Fig. 5b, and Fig. 5c show that the real stimuli tend to have more noise and do not adhere to a strong pattern when compared to Fig. 4a, Fig. 4b, and Fig. 4c. Table VI shows that the models do not perform as well on real world stimuli. ballRoll[1-3] is the simplest real stimulus, and as such **P** and **AP** achieved full accuracy. LGMD and **A** missed one roll.

Surprisingly good results come from the cupQUAV stimulus: 70% accuracy for all models except for **AP**, which had 80%. It is worth noting that **AP** performed consistently well when compared with the other models.

The real world stimuli tended to contain more activations due to the irregularity of the shapes and increased noise. The possibility of detecting the hand by stochastically dropping pixel-events, was investigated. Dropping 50% of the DVS events and re-optimising the network gave 100% accuracy for the hand and cupQuad stimulus. However, in doing this,

(a) Filtered ballRoll2 Input (P Layer Raster Plot).
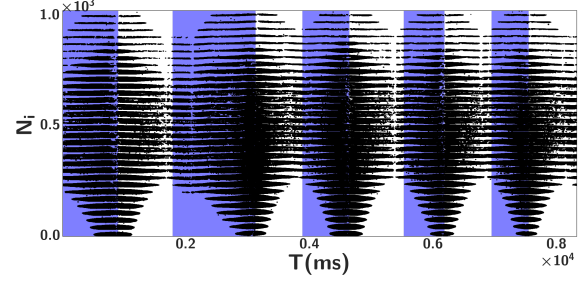

(b) Filtered cupQUAV Input (P Layer Raster Plot).
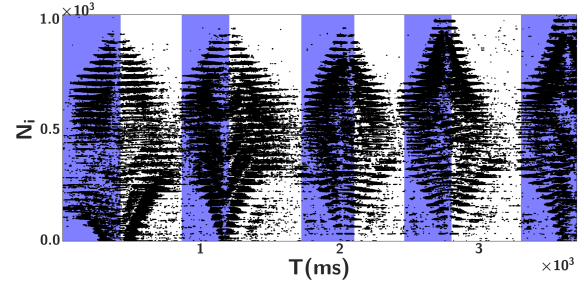
(c) Filtered Hand Input (P Layer Raster Plot).

Fig. 5: Complex real stimuli. The white and coloured backgrounds indicate non-looming and looming respectively.

TABLE VI: Quality metrics of the performance of different LGMD models for different real looming stimuli

| Stim	Model	Acc	Sen	Pre	Spe
ballRoll[1-3]	**LGMD**	0.66	0.66	1.00	0.00
	A	0.66	0.66	1.00	0.00
	P	1.00	1.00	1.00	0.00
	AP	1.00	1.00	1.00	0.00
cupQUAV	**LGMD**	0.70	1.00	0.62	0.40
	A	0.70	1.00	0.62	0.40
	P	0.70	1.00	0.62	0.40
	AP	0.80	1.00	0.71	0.60
hand	**LGMD**	0.50	1.00	0.50	0.00
	A	0.50	1.00	0.50	0.00
	P	0.50	1.00	0.50	0.00
	AP	0.50	1.00	0.50	0.00

the network was no longer robust to the speed changes in the composite benchmark test. Indeed, even using all of the pixels, the network could be optimised to work on the real world stimuli, but would no longer be as accurate for inputs that contained less activations. The inhibition values went up and the gain values went down, meaning the network struggled to spike on stimuli that weren't noisy or event heavy. Some sort of additional pre-filtering could be useful in getting the looming network to be fully robust in all situations, such as a sub-sampling filter that allows less than some maximum number of input pixels to be active at a given time stepls to be active at a given time step.
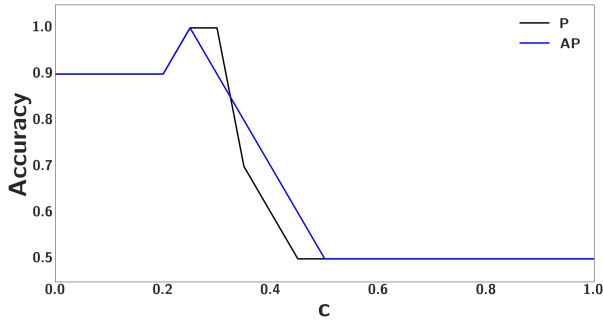
(a) Effect of changing the c clamping value on the learning weight w for the composite stimulus
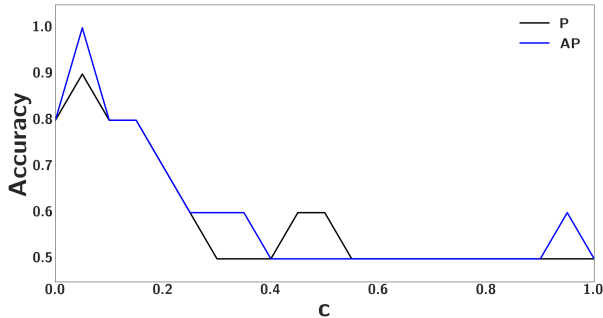
(b) Effect of changing the c clamping value on the learning weight w for the circleSlow stimulus
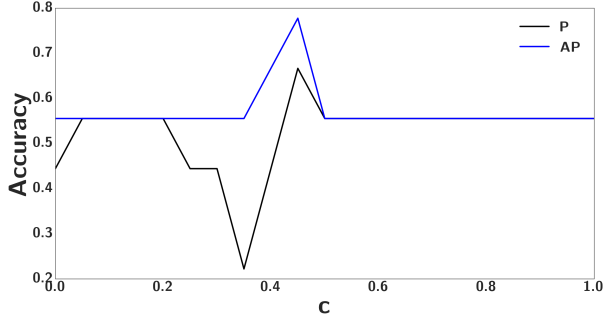
(c) Effect of changing the c clamping value on the learning weight w for the hand stimulus

Fig. 6: The effect of changing the clamping value on various stimuli

1) The effect of changing c on plasticity: Fig. 6a, Fig. 6b, and Fig. 6c show how changing the bounds of the plasticity clamping changes the LGMD (**P** model) accuracy for the composite, cicleSlow, and hand stimuli respectively.

Interestingly, for the two simulated stimuli increasing the clamping to beyond 25% caused the accuracy to drop to 50%. The sensitivity dropped to 0% indicating that it was no longer detecting looms and that the synaptic weights were no longer causing the LGMD neuron to fire.

Increasing the clamping to 45% increases the accuracy for both the **P** and **AP** models on the hand stimulus. This shows that plasticity is a double edged sword that can both improve and degrade the performance of the model. In the simulated stimuli lower clamping, $c = 0.1 - 0.2$, tended to perform better but in the hand stimulus larger, $c = 0.5$, gave better results. This could suggest that too much plasticity can cause the weights to deviate too far from their good values on well formed stimuli but help to reduce noise in real world stimuli. Knowledge about the nature of your input can help to determine what level of plasticity you require. In all simulated and real cases except for the hand stimulus, a small contribution of plasticity improved the performance.

VI. CONCLUSIONS

We implemented a neuromorphic model of the locust LGMD network using recordings from a UAV equipped with a DVS sensor as inputs. The neuromorphic LGMDNN was capable of differentiating between looming and non-looming stimuli. It was capable of detecting the black and white simple stimuli correctly regardless of speed and shape. Real-world stimuli performed relatively well using the parameters found by the optimiser for synthesised stimuli. However, when re-optimised, the real-world stimuli performed comparably to the synthesised stimuli. This was mainly because real-world stimuli tend to contain a higher number of luminance changes and therefore the magnitude parameters needed to be reduced.

We showed that BO, DE, and SADE are capable of finding parameter values that give the desired performance in the LGMDNN model. It can be seen that SADE statistically significantly outperformed DE on all metrics besides specificity and the number of evaluations, although the only metrics that formed part of the objective function were fitness and accuracy. Once a suitable objective function was found that accurately described the desired output of the LGMDNN, BO, DE and SADE outperformed hand-crafted attempts, but a uniform random search also performed well. The algorithms were able to achieve 100% accuracy on black and white simple stimuli of varying shapes and speeds. SADE performed well in this task and we have shown that it is suitable for the optimisation of a multi-layered LGMD spiking neural network. This could save time when developing biologically plausible SNNs in related applications.

We have also studied effect of synaptic plasticity and neuronal spike-frequency adaptation on the performance of the LGMDNN, using the most successful parameter optimisation method. Our conclusion is that plasticity plays an important role in increasing (and decreasing) performance, depending on how its parameters are selected.

In the future, we plan to apply the optimisation algorithms directly to tuning the neuromorphic processors implementation of the model, with the end goal being a closed loop control

system on a UAV. Showing that the optimisation approach proposed is effective for selecting parameters directly on closed-loop neuromorphic hardware set-ups will greatly increase their usability.

REFERENCES

[1] G. Loianno, D. Scaramuzza, and V. Kumar, "Special issue on highspeed visionbased autonomous navigation of uavs," *Journal of Field Robotics*, vol. 35, no. 1, pp. 3–4. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21773

[2] S. C. Liu and T. Delbruck, "Neuromorphic sensory systems," *Current Opinion in Neurobiology*, vol. 20, no. 3, pp. 288–295, 2010.

[3] E. Chicca, F. Stefanini, C. Bartolozzi, and G. Indiveri, "Neuromorphic electronic circuits for building autonomous cognitive systems," *Proceedings of the IEEE*, vol. 102, no. 9, pp. 1367–1388, 2014.

[4] G. Indiveri, F. Corradi, and N. Qiao, "Neuromorphic architectures for spiking deep neural networks," in *2015 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2015, pp. 4–2.

[5] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. H. Weng, A. Wild, Y. Yang, and H. Wang, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, January 2018.

[6] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)," *Biomedical Circuits and Systems, IEEE Transactions on*, pp. 1–17, 2017. [Online]. Available: http://ncs.ethz.ch/pubs/pdf/Moradi_etal17.pdf

[7] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug 2014. [Online]. Available: http://www.sciencemag.org/content/345/6197/668

[8] S. Furber, F. Galluppi, S. Temple, and L. Plana, "The SpiNNaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, May 2014.

[9] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128x128 120 dB 15 μs latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb 2008.

[10] T. Serrano-Gotarredona and B. Linares-Barranco, "A 128×128 1.5% contrast sensitivity 0.9% FPN 3 μs latency 4 mW asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 3, pp. 827–838, 2013.

[11] T. Delbruck, "Frame-free dynamic digital vision," in *Proceedings of Intl. Symp. on Secure-Life Electronics, Advanced Electronics for Quality Life and Society*, 2008, pp. 21–26.

[12] E. Mueggler, B. Huber, and D. Scaramuzza, "Event-based, 6-dof pose tracking for high-speed maneuvers," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2761–2768.

[13] M. Müller, S. Lupashin, and R. D'Andrea, "Quadrocopter ball juggling," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 5113–5120.

[14] D. Brescianini, M. Hehn, and R. D'Andrea, "Quadrocopter pole acrobatics," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 3472–3479.

[15] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2520–2525.

[16] M. Blanchard, F. C. Rind, and P. F. Verschure, "Collision avoidance using a model of the locust LGMD neuron," *Robotics and Autonomous Systems*, vol. 30, no. 1, pp. 17–38, 2000.

[17] S. Yue and F. C. Rind, "A collision detection system for a mobile robot inspired by the locust visual system," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2005, no. April, pp. 3832–3837, 2005.

[18] S. Yue, F. C. Rind, M. S. Keil, J. Cuadri, and R. Stafford, "A bio-inspired visual collision detection mechanism for cars: Optimisation of a model of a locust neuron to a novel environment," *Neurocomputing*, vol. 69, no. 13-15, pp. 1591–1598, 2006.

[19] S. Yue and F. C. Rind, "Collision detection in complex dynamic scenes using an LGMD-based visual neural network with feature enhancement," *IEEE Transactions on Neural Networks*, vol. 17, no. 3, pp. 705–716, 2006.

[20] C. Hu, F. Arvin, C. Xiong, and S. Yue, "A Bio-inspired Embedded Vision System for," *IEEE Transactions on Autonomous Mental Development*, vol. 9, no. 3, p. 241, 2017.

[21] M. Hartbauer, "Simplified bionic solutions: A simple bio-inspired vehicle collision detection system," *Bioinspiration and Biomimetics*, vol. 12, no. 2, 2017.

[22] F. C. Rind, S. Wernitznig, P. Pölt, A. Zankel, D. Gütl, J. Sztarker, and G. Leitinger, "Two identified looming detectors in the locust: ubiquitous lateral connections among their inputs contribute to selective responses to looming objects," *Scientific reports*, vol. 6, p. 35525, 2016.

[23] R. D. Santer, R. Stafford, and F. C. Rind, "Retinally-generated saccadic suppression of a locust looming-detector neuron: investigations using a robot locust," *Journal of The Royal Society Interface*, vol. 1, no. 1, pp. 61–77, 2004.

[24] S. Yue, R. D. Santer, Y. Yamawaki, and F. C. Rind, "Reactive direction control for a mobile robot: a locust-like control of escape direction emerges when a bilateral pair of model locust visual neurons are integrated," *Autonomous Robots*, vol. 28, no. 2, pp. 151–167, 2010.

[25] R. Stafford, R. D. Santer, and F. C. Rind, "A bio-inspired visual collision detection mechanism for cars: combining insect inspired neurons to create a robust system," *BioSystems*, vol. 87, no. 2, pp. 164–171, 2007.

[26] L. Salt, G. Indiveri, and S. Yulia, "Obstacle avoidance with LGMD neuron : towards a neuromorphic UAV implementation ." in *International Symposium on Circuits and Systems*, 2017.

[27] F. C. Rind and D. Bramwell, "Neural network based on the input organization of an identified neuron signaling impending collision," *Journal of Neurophysiology*, vol. 75, no. 3, pp. 967–985, 1996.

[28] R. Brette and W. Gerstner, "Adaptive exponential integrate-and-fire model as an effective description of neuronal activity," *Journal of neurophysiology*, vol. 94, no. 5, pp. 3637–3642, 2005.

[29] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, and G. Indiveri, "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses," *Frontiers in Neuroscience*, vol. 9, pp. 1–17, 2015.

[30] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[31] R. Dong, "Differential evolution versus particle swarm optimization for pid controller design," in *Natural Computation, 2009. ICNC '09. Fifth International Conference on*, vol. 3, Aug 2009, pp. 236–240.

[32] N. Karaboga and B. Cetinkaya, "Performance comparison of genetic and differential evolution algorithms for digital fir filter design," in *Advances in Information Systems*, ser. Lecture Notes in Computer Science, T. Yakhno, Ed. Springer Berlin Heidelberg, 2005, vol. 3261, pp. 482–488.

[33] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE transactions on evolutionary computation*, vol. 15, no. 1, pp. 4–31, 2011.

[34] M. E. H. Pedersen, "Good parameters for differential evolution," *Magnus Erik Hvass Pedersen*, 2010.

[35] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE transactions on evolutionary computation*, vol. 10, no. 6, pp. 646–657, 2006.

[36] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[37] G. D. Howard, "On self-adaptive mutation restarts for evolutionary robotics with real rotorcraft," in *Proceedings of the 17th Annual Conference on Genetic and Evolutionary Computation*. ACM, 2017, p. In press.

[38] E. Brochu, V. M. Cora, and N. De Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv preprint arXiv:1012.2599*, 2010.

[39] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[40] G.-Q. Bi and M.-M. Poo, "Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type." *J. Neurosc*, vol. 77, no. 1, pp. 551–555, 1998.

[41] G. Howard, E. Gale, L. Bull, B. de Lacy Costello, and A. Adamatzky, "Evolution of plastic learning in spiking networks via memristive connections," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 5, pp. 711–729, 2012.

[42] P. Turney, D. Whitley, and R. W. Anderson, "Evolution, learning, and instinct: 100 years of the baldwin effect," *Evolutionary Computation*, vol. 4, no. 3, pp. iv–viii, 1996.

[43] L. Liu, C. Shen, and A. van den Hengel, "The treasure beneath convolutional layers: Cross-convolutional-layer pooling for image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4749–4757.

[44] A. Babenko and V. Lempitsky, "Aggregating local deep features for image retrieval," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1269–1277.

[45] B. Fernando, E. Gavves, J. Oramas, A. Ghodrati, and T. Tuytelaars, "Rank pooling for action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 4, pp. 773–787, 2017.

[46] S. Song, K. D. Miller, and L. F. Abbott, "Competitive hebbian learning through spike-timing-dependent synaptic plasticity," *Nature neuroscience*, vol. 3, no. 9, pp. 919–926, 2000.

[47] J. Sjöström and W. Gerstner, "Spike-timing dependent plasticity," *Spike-timing dependent plasticity*, p. 35, 2010.

[48] J. Mockus, "Application of bayesian approach to numerical methods of global and stochastic optimization," *Journal of Global Optimization*, vol. 4, no. 4, pp. 347–365, 1994.

[49] D. R. Jones, "A taxonomy of global optimization methods based on response surfaces," *Journal of global optimization*, vol. 21, no. 4, pp. 345–383, 2001.

[50] M. J. Sasena, "Flexibility and efficiency enhancements for constrained global design optimization with kriging approximations," Ph.D. dissertation, General Motors, 2002.

[51] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[52] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," *arXiv preprint arXiv:0912.3995*, 2009.

[53] D. J. Lizotte, *Practical bayesian optimization*. University of Alberta, 2008.

[54] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[55] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, 2001.

[56] D. F. Goodman, M. Stimberg, P. Yger, and R. Brette, "Brian 2: neural simulations on a variety of computational hardware," *BMC Neuroscience*, vol. 15, no. 1, p. 1, 2014.

[57] L. Salt, "Optimising a Neuromorphic Locust Looming Detector for UAV Obstacle Avoidance," Master's thesis, The University of Queensland, 2016.

[58] E. Alpaydin, *Introduction to Machine Learning.*, ser. Adaptive Computation and Machine Learning. The MIT Press, 2014, vol. Third edition.
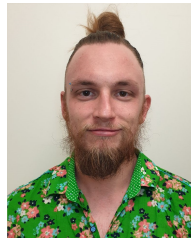
Llewyn Salt is a PhD candidate in Reinforcement Learning at the University of Queensland, Australia. His research is focused on utilising goals in continuous control problems in reinforcement learning.

He received a bachelor's (first class honours) and master's degree in mechatronic engineering from the University of Queensland. The work in this paper was undertaken at the Institute of Neuroinformatics at the University of Zurich and ETH Zurich as part of his master's degree program.

David Howard David is a Senior Research Scientist at CSIRO, Australia's national science body. He works and leads multiple projects at the intersection of robotics, evolutionary machine learning, and robotic materials. His interests include nature-inspired algorithms, learned autonomy, soft robotics, the reality gap, and evolution of form. His work has been featured in local and national media, TechXplore, and Wired.

He received his BSc in Computing from the University of Leeds in 2005, and completed a MSc in Cognitive Systems at the same institution in 2006. In 2011 he received his PhD from the University of the West of England. He is a member of the IEEE and ACM, and an avid proponent of education, STEM, and outreach activities.

Giacomo Indiveri Giacomo Indiveri is a Professor at the University of Zurich and ETH Zurich, Switzerland. He obtained an M.Sc. degree in Electrical Engineering and a Ph.D. degree in Computer Science from the University of Genoa, Italy. Indiveri was a post-doctoral research fellow in the Division of Biology at the California Institute of Technology (Caltech) and at the Institute of Neuroinformatics of the University of Zurich and ETH Zurich. He was awarded three ERC grants and is an IEEE Senior member. His research interests lie in the study of real and artificial neural processing systems, and in the hardware implementation of neuromorphic cognitive systems, using full custom analog and digital VLSI technology.

Yulia Sandamirskaya is a Group Leader in the Institute of Neuroinformatics (INI) at the University of Zurich and ETH Zurich. Her group Neuromorphic Cognitive Robots develops neuro-dynamic architectures for embodied cognitive agents. In particular, she studies memory formation, motor control, and autonomous learning in spiking and continuous neural networks, realised in neuromorphic hardware interfaced to robotic sensors and motors. She has a degree in Physics from the Belarussian State University in Minsk, Belarus and Dr. rer. nat. from the Institute for Neural Computation in Bochum, Germany. She is the chair of EUCOG the European Society for Cognitive Systems and the coordinator of the NEUROTECH project that supports and develops the neuromorphic computing community in Europe.