Perception Understanding Action: Adding Understanding to the Perception Action Cycle with Spiking Segmentation

Paul Kirkland [1,*], **Gaetano Di Caterina** [1], **John Soraghan** [1] **and George Matich** [2]

[1] *Neuromorphic Sensor Signal Processing Lab, Centre for Image and Signal Processing, Electrical and Electronic Engineering, University of Strathclyde, Glasgow, Scotland, UK*
[2] *Leonardo MW Ltd, London, UK*

Correspondence*:
Paul Kirkland
paul.kirkland@strath.ac.uk

ABSTRACT

Traditionally the Perception Action cycle is the first stage of building an autonomous robotic system and a practical way to implement a low latency reactive system within a low Size, Weight and Power (SWaP) package. However, within complex scenarios, this method can lack contextual understanding about the scene, such as object recognition-based tracking or system attention. Object detection, identification and tracking along with semantic segmentation and attention are all modern computer vision tasks in which Convolutional Neural Networks (CNN) have shown significant success, although such networks often have a large computational overhead and power requirements, which are not ideal in smaller robotics tasks. Furthermore, cloud computing and massively parallel processing like in Graphic Processing Units (GPUs) are outside the specification of many tasks due to their respective latency and SWaP constraints. In response to this, Spiking Convolutional Neural Networks (SCNNs) look to provide the feature extraction benefits of CNNs, while maintaining low latency and power overhead thanks to their asynchronous spiking event-based processing. A novel Neuromorphic Perception Understanding Action (PUA) system is presented, that aims to combine the feature extraction benefits of CNNs with low latency processing of SCNNs. The PUA utilises a Neuromorphic Vision Sensor for Perception that facilitates asynchronous processing within a Spiking fully Convolutional Neural Network (SpikeCNN) to provide semantic segmentation and Understanding of the scene. The output is fed to a spiking control system providing Actions. With this approach, the aim is to bring features of deep learning into the lower levels of autonomous robotics, while maintaining a biologically plausible STDP rule throughout the learned encoding part of the network. The network will be shown to provide a more robust and predictable management of spiking activity with an improved thresholding response. The reported experiments show that this system can deliver robust results of over 96% and 81% for accuracy and Intersection over Union, ensuring such a system can be successfully used within object recognition, classification and tracking problem. This demonstrates that the attention of the system can be tracked accurately, while the asynchronous processing means the controller can give precise track updates with minimal latency.

Keywords: Spiking, Convolution, Segmentation, Tracking, STDP, Neuromorphic, Neural Network, Asynchronous

1 INTRODUCTION

Understanding and reasoning is a fundamental process in most biological perception action cycles. It is through understanding of our visual perception that helps to inform our basic decision-making processes like ‘friend or foe” and “edible or inedible”, which ultimately is key to progression or survival. Adding some level of understanding into this cycle can help to deliver a robust robotic system that could perform more complex variations of simple following and tracking tasks. Computer Vision (CV) has made this understanding a reality for robotics systems, with traditional CV methods providing simple feature extraction at low latency, or modern deep learning-based Convolutional Neural Networks (CNN) providing state of the art results in almost every task with high precision and accuracy, but at the cost of higher latency and computation throughput. This often leaves the CNN out of the reach of the small robotic system world due to its lower power and computational specifications. Modern research looks towards biological inspirations to help solve these tasks, by bringing forward neuromorphic robotics, which seeks to merge the computational advantages of system such as the neuromorphic event-based vision sensor (NVS) and neuromorphic processors together, combined with Spiking Neural Network (SNN) which can allow for processing and control system structures. Typically a robotic system in this domain might aim to reach a Perception, Cognition, Action cycle, while the simpler approach of Understanding as a step toward cognition could be realised in an easier way, using the Perception Understanding Action (PUA) cycle as a stepping stone towards this goal.

Perception using neuromorphic vision sensors has become a promising solution. An NVS, as for example the Dynamic Vision Sensor (DVS) (Lichtsteiner et al., 2008), mimics the biological retina to generate spikes in the order of microseconds, in response to the pixel-level changes of brightness caused by motion. NVSs offer significant advantages over standard frame-based cameras, with no motion blur, a high dynamic range, and latency in the order of microseconds (Gehrig et al., 2018). Hence, the NVS is suitable for working under poor light conditions and on high-speed mobile platforms. There has been considerable research detailing the advantages of using an NVS in various vision tasks, such as high-speed target tracking (Mueggler et al., 2017; Lagorce et al., 2015) and object recognition (Kheradpisheh et al., 2018). Moreover, due to the fact that a pixel of an NVS is a silicon retinal neuron represented by an asynchronously generated spiking impulse, this can be directly fed into Spiking Neural Networks (SNNs) as input spikes for implementing target detecting and tracking in a faster and more neuromorphic approach.

Understanding through asynchronous spiking event-based computations like SNNs, often seen as the low latency biologically inspired alternative to CNNs, could provide an alternative solution to tracking and segmentation problems, through the ability to only compute on the currently active parts of the network, which in comparison to Artificial Neural Networks (ANN) and CNNs can require orders of magnitude less power consumption (Park et al., 2014). SNNs differ from normal computation processing and take inspiration from closer to biology, where expensive memory access operations are negated due to computations and memory being exclusively local (Paugam-Moisy and Bohte, 2012). Instead of using numerical representations like traditional methods, SNNs use spikes to transmit information with a key emphasis on the timing of those spikes. Several methods exist to train SNNs, with recent implementations seeing a conversion from CNN to SNN (Cao et al., 2015; Hunsberger and Eliasmith, 2015; Sengupta et al., 2019; Kim et al., 2019) yield promising results and open SNN architectures to the wider Machine and Deep Learning (ML-DL) audience. However, this method is still burdened with the training computational overhead and does little to utilise the efficiency of event driven computations. The SNN’s Spike Time Dependent Plasticity (STDP) and spike-based back-propagation learning have been demonstrated to capture hierarchical features in SpikeCNNs (Kheradpisheh et al., 2018; Masquelier and Kheradpisheh, 2018; Masquelier and Thorpe, 2007; Panda et al., 2017; O'Connor et al., 2013; Falez et al., 2019). Both of these methods better equip the network to deal with event driven sensors, where the significant gains over CNNs could be realised.

This work aims to build on the already successful perception-action models (Xie, 2003; Masuta et al., 2017; Bohg et al., 2017; Nishiwaki et al., 2003) and add some semantic understanding to the robotic system. With image segmentation seen as a critical low-level visual routine for robot perception, a semantic understanding of the scene can play an important role for robots to understand the context in their operational environment. This context can then lead to a change in the action that could be undertaken. In this article, we show how using a spiking fully convolution neural network for event-based segmentation of a neuromorphic vision sensor can lead to accurate perception and tracking capabilities with low latency and computation overhead. Leveraging this spiking event-based segmentation framework to feed a spiking control system allows the low latency to continue from the perception to the action.

The PUA system presented builds on SpikeSEG, a spiking segmentation network from previous work (Kirkland et al., 2020), and extends it with a systematic approach to spike-based object recognition with tracking, lateral inhibition classifications, a new thresholding mechanism and modification to STDP learning process. Moreover, differently from (Kirkland et al., 2020), the novel work presented is applied to a different application context, i.e. object recognition with attention. In light of this the novel contributions of this work include:

- SpikeSEGs segmentation output is integrated into a spike-based control system to produce the Perception-Understanding-Action system where the segmentation infers the attention of the system to allow controller track updates.
- The revised network includes more features to enhance the segmentation ability, including:
  - Lateral inhibition pseudo classification mechanism for semantic segmentation-based attention.
  - A new Pre-Empt then Adapt Thresholding (PEAT) approach designed to deal with potentially noisy, corrupt or adversarial inputs.
  - A modification to the STDP learning rules to include feature pruning (resetting) if under/over utilised.

The rest of the paper is organized as follows. Section 2 reviews related research topics covering each of the PUA framework individual sections. Section 3 presents the methodology, with an insight to each of the proposed system components. The results are detailed in section 4 and section 5 provides the conclusion.

2 RELATED WORK

The allure of low latency object recognition and localisation has brought the attractive features of the NVS (mainly the DVS) to the forefront of research. Early low latency control examples, such as the Pencil Balancer (Conradt et al., 2009) and the Robotic Goalie (Delbruck and Lang, 2013), help to highlight the latency advantages that an NVS can provide. Exploiting the sparse and asynchronous output of the sensor allow successful applications to these low latency reactive tasks. However, both systems fall short of fully capitalising on the event-driven asynchronous output, through a processing and control regime of similar nature.

The concept of exploiting the NVS low latency continues into object tracking. Low latency tracking relies upon robust feature detection, with geometric shapes being ideal features to detect. A number of methods have been implemented successfully, such as geometric constraints (Clady et al., 2015) along with advanced corner detection methods, as for example Harris (Vasco et al., 2016) and FAST (Mueggler et al., 2017). The use of more complex features such as Gaussians, Gabors and other hand crafted kernels (Lagorce et al., 2015) provides a pathway to modern Convolutional Neural Network feature extraction approaches (Li and Shi, 2019), that implement a correlation filter from the learned features of the CNN. This allows a multi-level approach whereby correlations of intermediate layers can also be performed to improve the inherent latency disadvantage of the CNN approach, albeit with an accuracy trade-off.

Spiking Neural Networks have seen success with NVS data used for object detection and classification (Bichler et al., 2012; Stromatias et al., 2017; Paulun et al., 2018). Recent work has implemented Spiking Convolutional Neural Networks (Kheradpisheh et al., 2018; Falez et al., 2019) with NVS-like data created using a difference of Gaussian filter, suggesting the combination of SNNs and Deep Learning could yield successful results (Tavanaei et al., 2019). SNNs have also been utilised for tracking with an NVS through implementations inspired by the Hough Transform (Wiesmann et al., 2012; Seifozzakerini et al., 2016; Jiang et al., 2019), to be able to detect and track lines and circles. Spiking Neural Networks can also be utilised to implement control systems, from simple altitude control (Levy, 2020) to an adaptive robotic arm controller (DeWolf et al., 2016). Ultimately the majority of research only utilises one aspect of the SNN, either processing or control. Even though SNNs have been shown to implement a full perception cognition action cycle with Spaun (Eliasmith et al., 2012), underpinning the ideology of a fully spike-based neuromorphic system similar to that proposed with the Perception Understanding Action framework in this paper.

3 METHODOLOGY

3.1 Perception-Understanding-Action Framework

The Perception-Understanding-Action framework specifies how the system will utilise the asynchronous event driven nature of the Neuromorphic spiking domain, and it is illustrated in Figure 1. In the Perception block, the NVS is used to sparsely and asynchronously encode the luminosity changes within the scene. In the Understanding block, inputs are understood through the use of the Encoder-Decoder SpikeCNN (SpikeSEG (Kirkland et al., 2020)) contextualising and building understanding of the scene through semantic segmentation. In the Action block, the segmented output is used to provide an input to the spike counters at the edge of the field of view, allowing a simplistic semantic tracking controller to be realised. This control output would then be able to influence motors or actuators to allow an asynchronous end to end Neuromorphic system. This system aims to provide a low latency competitor to the Perception Action robotic system where the sensor input is directly fed to the controller, while providing an upgraded feature representation to the more complex line and edge detection-based approaches. The system can even provide benefits or replace some computer vision-based robotic tasks which utilise CNNs for complex feature extraction, while providing lower latency and computational overhead. Furthermore, compared to the CNN, the SCNN provides a more readily understandable processing stage, where features are sparse and more visually interpretable.

3.2 Perception

A key element in producing a low latency system with a low computational overhead is to have a sensor that can exploit the sparse and asynchronous computational elements of an SNN while still giving a detailed recording of the scene. Neuromorphic Vision Sensors (NVS) *(event-based Vision Sensors)* (Lichtsteiner et al., 2008; Brandli et al., 2014) have recently become more popular and widespread. These camera-like devices are bio-inspired vision sensors that attempt to emulate the functioning of biological retinas. They differ from conventional cameras in that, they don't record all the information the sensor sees at set intervals. Instead these sensors produce an output only when a change is detected. This in turn means they are

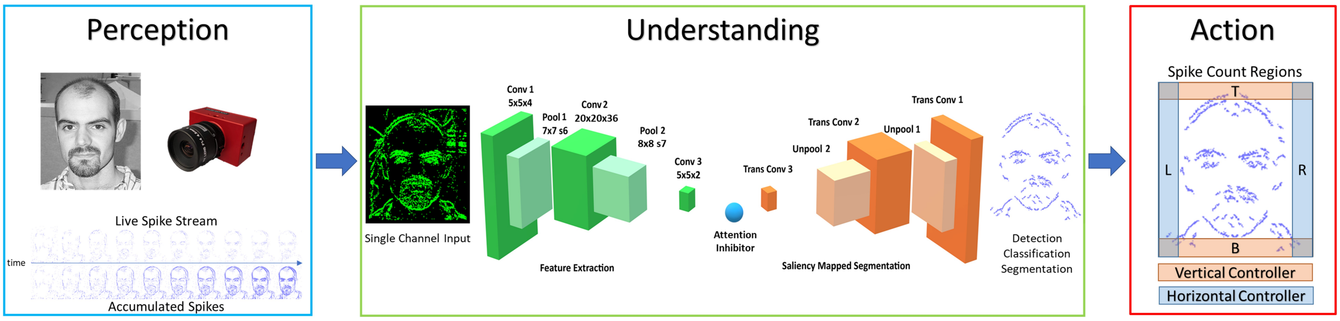


Figure 1. Perception Understanding Action Framework, with internal system diagrams showing the Perception input (image from Caltech Dataset (Li Fei-Fei et al., 2018) , the Understanding network SpikeSEG (Kirkland et al., 2020) and the Action controller method.

capturing the luminosity at a set point in time, meaning a continuous temporal derivative of luminosity is output. Whenever this happens, an event $e = [x, y, ts, p]$ is created, indicating the x and y position along with the time ts at which the change has been detected and its polarity, where $p \in \{1, -1\}$ is a positive or negative change in brightness. This change in operation not only increases the sparsity of the signal but allows for it to output asynchronously. Resulting in microsecond temporal resolution and considerably lower power consumption and bandwidth. These parameters make the NVS an ideal candidate for object tracking, especially of fast moving objects (Delbruck and Lichtsteiner, 2007; Glover and Bartolozzi, 2017), however many methods are still yet to utilise this spiking sensor within a match spiking processing such as SNNs.

3.3 Understanding through Spiking Segmentation

The Understanding of this system is inferred from the semantic segmentation operation carried out by the SpikeSEG network (Kirkland et al., 2020), seen in Figure 1 within the Understanding block. The SpikeSEG segmentation network has received a number of improvements and upgrades along with its integration within the PUA framework.

3.3.1 Network Architecture

The network architecture illustrated within Figure 1 (Understanding) is made up of two main sections seen in green and orange, that relate to the encoding and decoding layers respectively. The network is split into these two sections where training only occurs on the encoding side, while the weights are tied to the mirrored decoding layers. This allows a integrate and fire neuron with layer-wise STDP mechanism with adaptive thresholding and pruning to be used to help compress the representation of the input to allow the decoding layer to segment the image based on the middle pseudo classification layers. This encoding-decoding structure symbolises a feature extraction then shape generation process. The learning of the encoding process aims to extract common spatial structures as useful features, then decode those learned features over to the shape generation process, unravelling the latent space classification representation but with a reduction in spike due to the max pooling process. The network has 9 computational layers *(Conv1-Pool1-Conv2-Pool2-Conv3-TransConv3-UnPool2-TransConv2-UnPool1-TransConv1)* as seen in Figure 1. Between the Conv3 and TransConv3 layers, there is a user-defined attention inhibition mechanism, which can operate in two manners: No Inhibition, which allows semantic segmentation of all recognised classes from the pseudo classification layer; or With Inhibition, that only allows one class to propagate

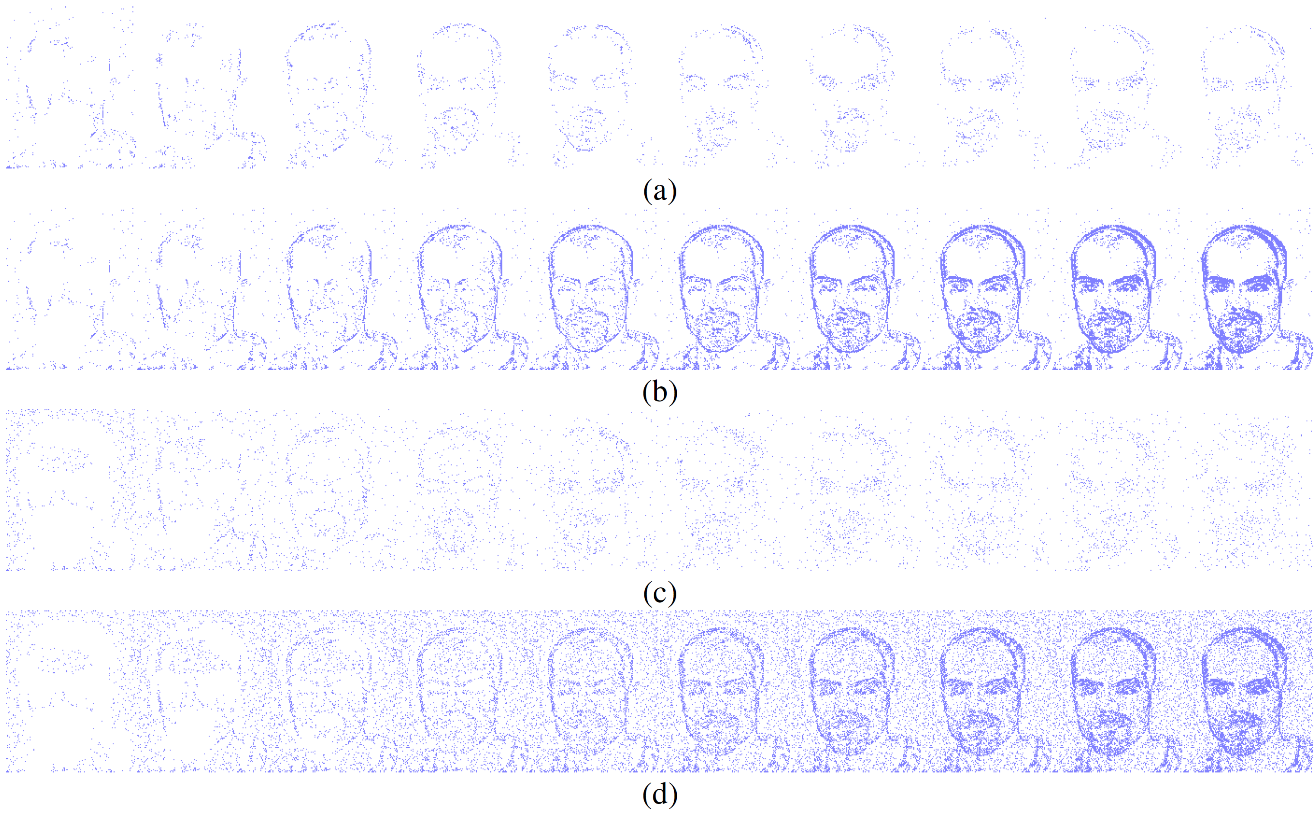


Figure 2. Input event streams from N-Caltech Dataset 'Face', with (a-b) showing a 10ms clip over 10 steps going from left to right. (a) showing the input to the network per step and (b) showing the accumulated inputs for easier visualisation. (c-d) show a 10ms clip over 10 steps with additive noise to show how extra noise affects the input stream, with (c) showing per step and (d) showing accumulated.

forward to the decoding layers. This attention not only provides a reduction in the amount of computation, but also simplifies the input to the controller.

3.3.2 Encoding

The encoding part of this system is derived from a basic SpikeCNN with a simplified STDP learning mechanism (Kheradpisheh et al., 2018). To allow the network to better suit the framework and encoding decoding structure a number of modification are applied. As the structure of the network is now fully convolutional there is no longer a requirement for a global pooling layer for classification. Instead the final convolution layer is utilised as a mock classifier by mapping the number of known classes to the number of kernel used for feature learning. This method is also used to help the interperitability of the system as having one kernel per classes allows for better visualisation of the network features. Through the use of a modified STDP rule and adaptive neuron thresholding, the encoder aims to capture the reoccurring features that are most salient through the event stream inputs. The input events are fed into the network via a temporal buffering stage, to allow for a more plausible current computing solution such as on the Intel Loihi Neuromorphic chip (Davies et al., 2018), while ideally they would just be a constant stream. To internally mimic the continuous data, 10ms of event data is buffered into 10 steps, representing 1ms each (this value of 10ms is chosen to empirical testing and based on the input spike count of the N-Caltech Dataset); this input data stream is shown in Figure 2. Fig 2, also illustrates what 1ms of data looks like over the 10ms (a) and how it looks if accumulated over 10ms (b). Figure 2 then demonstrates how added noise affects the input stream, repeating the images in Fig 2 (a) and (b) with noise in 1ms steps in (c) and

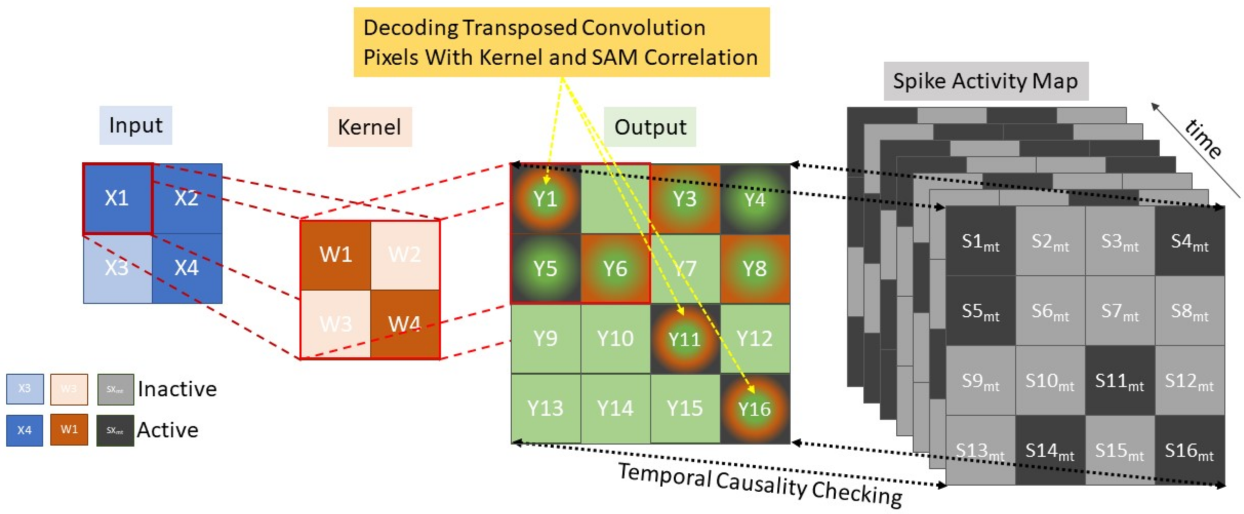


Figure 3. Decoding using transposed convolutions with spike activity mapping, resulting in active pixel saliency mapping

accumulated over 10ms in (d). For each time step in the encoding processing, a spike activity map Sk_{mt} is also produced, where *m* is the feature map and *t* is the time step. This allows an account of the exact spatial time location of each active pixel used in the encoding processing, which helps allow the decoder to map these active areas back into the pixel space.

3.3.3 Decoding

The Decoding Process makes use of the same unpooling and transpose convolutions as (Long et al., 2015; Simonyan et al., 2013; Badrinarayanan et al., 2017; Zeiler and Fergus, 2014) taking pixels in the latent classification space back into the original pixel space. However, no learning mechanism is used, as the mapping is based on temporal activity and pixel saliency mapping, utilising a similar method to tied weights (Hinton et al., 2006) and switches (activations within the pooling layers) from the encoding layer to map directly to the decoding such that $W_{ij(encoding)} = W_{ji(decoding)}$. This modification is required to deal with the temporal component of the spiking network, as now the latent pixel space representation must be unravelled with the constraints and context of space and time. Changes are made to both the transposed convolutions and the unpooling layers. The transposed convolution still functions as a fractionally strided convolution of the weight kernel as normal. However, now an extra step of comparing the output mapping with a temporal spike activity map of the post convolution pixel space is required as illustrated in Figure 3, where the conventional Input via Kernel to Output stage remains, with an added Spike Activity Map check on each term in the output for temporal causality.

Since the encoding neurons emit at most one spike per buffered time input, the Spike Activity Map is used to keep track of the first spike times (in time-step scale) of the neurons. Every stimulus is represented by M feature maps, each constitutes a grid of neurons seen as a kernel value K, equal to the row-major linear indexing of the kernel. Let Tp be the processing steps between the tied encoding and decoding layer with a maximum possible difference of 9 processing time-steps (5 encoding and decoding layers each). While each encoding layer has a value $Te_{m,k}$, which denotes the spike time of the encoding neuron placed at position (k) of the feature map m, where $0 \leq m < M, 0 \leq k < K$. The individual decoding layer then considers this stimulus as a three-dimensional binary spike tensor S of size $Tp_{max} \times M \times K$ where a spike in the decoding layer Sd is a function of :

$$Sd(Tp, Te, m, k) = \begin{cases} 1 & Td_{m,k} = Te_{m,k} + Tp \\ 0 & otherwise \end{cases} \tag{1}$$

Where the decoding time $Td_{m,k}$ for each map and kernel value is compared to the equivalent encoding layer $Te_{m,k}$ offset by the processing time Tp. It is this $Te_{m,k} + Tp$ that is represented by the Spike Activity Map shown in Figure 3 where $Sk_{m,t}$ is illustrated as the process ensuring $Td_{m,k} = Te_{m,k} + Tp$ while 'Output' demonstrates an example of the transposed convolution process. To reduce memory overhead only the last 9 Spike Activity Maps as this is the minimum requirement to ensure temporal causality. Within Figure 3, the green and orange squares represent the transposed convolution outputs and the green, orange and black outputs represent the outputs from the transposed convolution decoding that also matched up with encoding layer, through correlation with the Spike Activity Map. This demonstrates how the Spike Activity Map reduces the 'Output' values to only those with equivalent temporal values. The saliency mapping occurs within the unpooling layers which operate on a similar manner in order to keep temporal causality, but due to the max pooling operation working in reverse only one pixel per pooling kernel is processed. With reference to Figure 3, this would mean the orange kernel would only have one active square, which reduces the output significantly. The measure allows only the most salient features to propagate through the decoding layers, resulting in the segmentation with only those features that best fit the pseudo classification. A verbal illustration being, if there are 9 time steps between Conv-1 and TConv-1, while only 5 steps between Conv-2 and TConv-2 and 1 step between Conv-3 and TConv-3. So, if a spike occurs at time step 2 within Conv-1, the temporal check will only allow TConv-1 to allow a spike at that location at time step 11.

3.3.4 Adaptive Neuron Thresholding

The adaptive neuron thresholding used within this paper builds upon the Pre-Emptive Neuron Thresholding (Kirkland et al., 2019, 2020). Improvements are made by no longer solely relying on synaptic scaling from the input number of spikes as a means of homoeostasis. Although this was successful in stopping the progression of less structured noise features within the first convolution layer and structured noise when synaptic scaling was applied to all layers. Along with the structured noise filtering process, this homoeostasis rule also accidentally removes some of the less common desired features from propagating as discrimination between these and noise from input spike count is insufficient. The update to the algorithm sees an adaptive element in the form of intrinsic layer-wise synaptic scaling (a layer-wise spike counter) added to the thresholding parameter to potentially counter this less common feature removal. During training the thresholding is set as follows

$$V_{thr}(S_{in}, S_l) = \begin{cases} \frac{K_l}{4} & \text{for} \quad S_{in} < S_{in(min)} \\ \left.\begin{matrix} c + mV_{thr} + h^{-} & \text{for } S_l < H_l \\ c + mV_{thr} + h^{0} & \text{for } S_l = H_l \\ c + mV_{thr} + h^{+} & \text{for } S_l > H_l \end{matrix}\right\} & \text{for} \quad S_{in(min)} < S_{in} < S_{in(max)} \\ \frac{K_l}{2} & \text{for} \quad S_{in} > S_{in(max)} \end{cases} \tag{2}$$

Where V_{thr} is the neuron threshold, dependent on both the spiking input rate, S_{in}, and the layer-wise spike rate, S_l. m is gradient of the linear relationship between V_{thr} and S_{in}, with c being the y-intercept. h the homoeostasis offset is determined to be either positive, negative or zero dependent on the layer-wise spike count, S_l when compared to the set homoeostasis value H_l. While K_l is the convolution kernel size within that layer. The equation follows a piecewise function such that V_{thr} is described as $\{ V_{thr} \in \mathbb{N} \mid \frac{K_l}{4} < V_{thr} < \frac{K_l}{2}\}$. When the spike input rate S_{in} is within a normal range, the function is then defined by the bounded linear relationship with the homoeostasis offset. The values of h^{-}, h^{0}, h^{+} and H_l are set through empirical testing by monitoring the range of S_l and S_{in} values from the N-Caltech dataset.

Once training is complete and the features within the convolution kernels are known, the thresholding changes to take into account the size of the feature, as the range of threshold values might now be smaller than in the training stage. This modification changes the outer bounds of the threshold as shown

$$V_{thr}(S_{in}, S_l) = \begin{cases} \frac{F_{min}}{2} & \text{for} \quad S_{in} < S_{in(min)} \\ \left.\begin{array}{ll} c + mV_{thr} + h^{-} & \text{for } S_l < H_l \\ c + mV_{thr} + h & \text{for } S_l = H_l \\ c + mV_{thr} + h^{+} & \text{for } S_l > H_l \end{array}\right\} & \text{for} \quad S_{in(min)} < S_{in} < S_{in(max)} \\ F_{min} & \text{for} \quad S_{in} > S_{in(max)} \end{cases} \tag{3}$$

Where F_{min} is the smallest feature size within that layer. This parameter change ensure the threshold value does not exceed the smallest feature size, which would result in that neuron being unable to reach firing potential. In both cases the training and testing the input spike count S_{in} value affects the threshold for each input spike buffer, while the layer-wise spike count S_l is average over 10 inputs.

This allows a layer-wise adaptability dependent on the amount of spiking within the previous layer. The algorithm now permits a high volume of spiking activity at the input to be initially pre-emptively dealt with, ensuring a large amount of spiking activity does not reach the controller, causing an undesired response. Then adapting the thresholds to allow sufficient spiking activity ensures a smoother and more robust controller output of the system. The key element of this method is to ensure a more robust and predictable outcome when a noisy, corrupt or adversarial input is received. With this being more of a concern due to the system be asynchronous end to end, a high volume incoherent input could directly lead to a wild or undesired response from the controller. This approach errs on the side of caution with the sudden increase in input spikes being inhibited first, and then excited to a desired level, in contrast to a typical intrinsic response of allowing the activity, and then inhibiting to a desired response.

3.3.5 Changes to STDP training with active pruning

A simplified unsupervised STDP rule (Kheradpisheh et al., 2018; Bi and Poo, 1998) is used throughout the training process, including a Winner Take All (WTA) approach to STDP, that operates by only allowing one neuron (feature) in a neuronal map (feature map) to fire per time constant; this is viewed as an intra map competition. This WTA approach then moves onto the inter map inhibition, only allowing one spike to occur in any given spatial region, typically the size of the convolution kernel, throughout all the maps. As a result of these inhibition measures, two features can tend towards representing the same feature until such point where one becomes more active, while the other gets inhibited to the point of infrequent or no use. At this stage the feature representation has become obsolete and can be pruned or reset, allowing the opportunity to form another more useful feature. To capture this information the layer-wise training method make use of the training layers convergence values

$$C_l = \sum_k \sum_i \frac{w_{ki}(1 - w_{ki})}{n_{w_{ki}}} \tag{4}$$

Where C_l is the convergence score for the layer and w_{ki} is the ith synaptic weight of the kth convolution kernel. The $n_{w_{ki}}$ is the number of individual weights contained with the layer calculated by kernel size and the number of kernels in the previous and current layers, $n_{w_{ki}} = K \times k_{pre} \times k_{cur}$. The pruning function makes use of the convergence score that is typically used to indicate when training is complete, as the convergence tends to zero due to the weights tending to 0 or 1. Noticing that the layer-wise convergence is just a sum across all the kernels allows a modification to calculate the convergence across each kernel within that layer with respect to all previous maps.

$$C_{k_{cur}} = \sum_{k_{pre}} \sum_i \frac{w_{k_{pre}i}(1 - w_{k_{pre}i})}{n_{w_{k_{pre}i}}} \tag{5}$$

This new terms $C_{k_{cur}}$ allows monitoring of each kernel during the learning process, as previously mentioned obsolete kernels that learned similar features are less active, resulting in higher convergence numbers while maintaining a high spiking activity. The high spiking activity is due to the kernel maintaining the high starting weight value which are random values drawn from a normal distribution with the mean of $\mu = 0.8$ and standard deviation of $\sigma = 0.05$. However the kernel does not exhibit a feature that allows it to spike quick enough to receive a weight update from the STDP WTA rule. As the kernel had already started a convergence to a particular feature, once under-active it then attempts to convergence to another commonly occurring feature. However, the kernel often convergences to a useless feature representation that is unhelpful to the final result of the network. This pruning method, rather than simply removing the kernel, gives it the chance to learn a new feature from scratch by resetting the kernels weights. Thus allowing the best chance of convergence to a useful feature. This pruning process takes place once the convergence value of the layer C_l drops below the original starting value. As initially the weights are deconverging from the mean weight initialisation, before returning to the original convergence value on the way to zero. Once this milestone has been reached the pruning function in activated

$$Prune_{k_{cur}}(C_{k_{cur}}, C_l, S_k) = \begin{cases} 1 & \text{for} \quad C_{k_{cur}} > \bar{C}_l + 1\sigma_{C_l} \quad \text{and} \quad S_k > \bar{S}_l + 3\sigma_{S_l} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where $\bar{C}_l$ is the mean convergence for that layer, σ_{C_l} is the standard deviation of that layers values, S_k is the spike activity within an individual kernel. $\bar{S}_l$ is the mean spike count of that layer and σ_{S_l} is it standard deviation. If a kernel value has a convergence score higher than 1 STD from the mean while having a spiking activity 3 STD higher than the mean spike rate in that layer, the kernel is reset with the initial weight distribution. Since many of the kernels are already converging to useful features this newly reset kernel will convergence to a new unrepresented feature.

3.3.6 Latent Space Inhibition for Attention

In order to have the network change its focus or attention, the latent space pseudo classification layer also acts as an inhibition layer for this mechanism. This operates by inhibiting other neurons in that layer if a specific neurons feature is chosen to be the attention. This is an external mechanism to the network as otherwise, the network will give equal attention to the full scene and semantically segments all known objects within a scene. This allows a simplification of the output of the network fed to the controller, allowing the attention of the system to be narrowed to that particular pseudo-class. This segmentation-based attention can then be used to follow a given class dependent on the output of the controller. It operates between convolution layer 3 and trans-convolution layer 3 with the same principals as the inter map inhibition with the encoder, though now the spatial region is the whole latent space. This inhibition can also work autonomously where the pseudo-class with the most activity is the attention of the network, allowing the network to switch attention to known classes based on their prevalence within the scene.

3.4 Tracking with Attention

The Action part of the system with its spiking controller is directly influenced by the attention mechanism, as when no attention is chosen the controller acts on all the segmented data being output by the SpikeSEG network. This could cause unwanted control output if the scene contained more than one known class, as unknown classes should still be removed by the process. Once a class has been chosen as the attention, the segmentation output is reduced to only that class, as illustrated in Figure 1 (Action), which allows for simple spike counter controller to produce a more robust and reliable output. The reduction in information initially by the NVS which then further reduces through the semantic segmentation and attention, allow the implementation of this simple spike counter. This is due to the segmentation output only containing information relating to the attention of the network, the controllers task is just to keep this in the center of the field of view. The simplicity of the controller also allows it to take advantage of the asynchronous event-driven system to provide low latency tracking updates a key element of the system. However, if there was more than one instance of a class in a scene there is no way to separate the two instances, so tracking would be based off all instances of a class. Nevertheless, this system would make an improvement over the purely spiking activity tracking systems by adding some semantic context to the activity, while the simplified spike counter in this instance allows class based tracking could be enhanced with more complex spike tracking such as dynamic neural fields (Renner et al., 2019)

4 RESULTS

In this section, a series of experiments on individual and multi event-stream recordings are presented. The metric used in this paper is the Intersection over Union (IoU, also know as the Jaccard Index) to grade the segmentation, which guides the control system of the network and ultimately, with user choice, the Attention of the system. This metric was used due to the availability of the bounding box annotations within the subset of the N-Caltech dataset that was used within the experiments. The feasibility of the attention-based tracking is also encapsulated within the IoU value, though due to the small saccade movements of the camera within the N-Caltech dataset, it is infeasible to use this to highlight spike-based tracking. This is due to two issues throughout the movements. The first is the IoU value only receives a small change as the displacement is often less than 10 pixels. The second is that the occurrence of segmented spike activity in the controlled regions, is due to the tight field of view around the class in scene. This results in the testing of the Perception and Understanding system only with this data. To ensure testing of the full Perception, Understanding and Action system, two further experiments were carried out. The first with multi input streams on a large input space and the second using our own captured DVS data of a desk ornament with a hand held sensor. Lastly, the results sections show how the system is more robust and interpretable than alternative models, with the use of the Pre-Empt and Adapt Thresholding and the contour like sparse features within the weights of SpikeSEG.

Within these experiments the step time for any processing is now linked to the input time step, meaning internal propagation of spikes takes one step (or 1ms) per layer, resulting in a 11ms lag to get the segmented results. This allows for better visualisation of the asynchronous manner of the processing and control for each step. However, this does not reflect the actual processing time of the network which, given its complexity compared to similar models ran on neuromorphic hardware, would most likely be able to execute this task in real-time for the 1ms step, meaning a full pass through the network per input step. However, testing in this manner would not fully highlight the asynchronous advantage especially within a dynamic environment.

One further note is that throughout all the testing the features of Convolution Layer 1 are pre-set to best found features for initial edge detection, which results in a horizontal, vertical and two diagonal lines which can be see later in the Interpretablity Section 4.3.2 within Figure 14.

4.1 Perception to Understanding with Segmentation

Initially two subset classes from the N-Caltech dataset (Orchard et al., 2015) are used to evaluate the Understanding section of the system. On this single stream input typically only containing a singular class with variable amounts of background noise and clutter, the network is able to gain an accuracy of 96.8% within the pseudo classification layer and a 81% mean Intersection over Union score over each of the 10ms buffered input that resulted in successful segmentation, results are also shown in Table 1. This is an improvement on the single results seen within (Kirkland et al., 2020) of 92% and 67% for accuracy and IoU, with the improved feature creation allowing a more detailed representation allowing an improvement in both the accuracy and segmentation. The test results are based on training with 200 samples from the Face and Motorbike classes with another 200 used for testing. This number was limited as the ”Easy Faces” has just over 400 images and was converted into ”Faces” within the N-Caltech dataset with the ”Faces” category being removed. 400 images provided an equal training set between the Face and Motorbike classes. The images in Figure 4 shows how the segmentation process was completed firstly through encoding the event stream input through 3 convolution and two pooling layers Fig 4 (b-d and i-k), resulting in a sparse latent space representation used to provide a classification of this binary task Fig 4 (d and k). Fig 4, then shows how the classification locations are then mapped back onto the pixel space through the undoing of the 3 convolutions and 2 pooling layers Fig 4 (e-g and l-n). For illustrative purposes, both the face and motorbike are accumulations of the network activity according to 10ms input buffer and full propagation of spikes through the network. Each convolution process is shown, with pooling omitted, Convolution Layer 1 is shown in Fig 4 (b and i) while layer 2 Fig 4 (c and j) with (d and k) showing the third convolution also used as pseudo classification. Fig 4 (e and l) show the second transposed convolutional layer, named to mirror the encoding side, while Fig 4 (f and m) show transposed convolution 1 and Fig 4 (g and n) display the segmented outputs. This segmentation result is shown overlapped onto the input for two examples within Figure 5. The colours used within Figure 4 are linked to the corresponding feature that was activated in that layer with Fig 4 (c and j) showing different coloured features active for each the face and motorbike, with section 4.3 exploring what the feature maps contain. This output from the SpikeSEG network can feed directly into the spiking controller of the PUA system, guiding any movement that would be required to follow the attention of the system. Although the controller in this context is unable to operate due to the narrow field of view and limited movement, the Understanding section of the system does still capture this small saccade movement of the camera within the segmented output as seen in this overlapped output image, Figure 6 with (a) showing a downward and right shift of the segmented pixels over time, relating to the inverse movement performed by the camera, while Fig 6 (b) and (c) show the two further saccade movements. The segmentation also maintains an IoU value of above 0.7 throughout the movement, meaning the segmentation is of good quality throughout (0.5 being acceptable, 0.7 being good and 0.9 being precise) (Zitnick and Dollár, 2014), for reference if the full input size is used for IoU the average output is approximately 0.57. Consequently, this means tracking would still be possible through alternative non-spiking methods such bounding boxes or centroid/center of mass calculation, but would remove the all spiking asynchronous feature of this system.

4.1.1 N-Caltech Dataset Extended

To further evaluate the scalability of the model, a further 2 experiments are carried out with 5 and 10 classes. This allowed testing the model with 2, 5 and 10 classes within the same experimental parameters , that being 16 features per class in second convolution layer and 1 per class in the third convolution layer, with active thresholding and pruning. 16 features was found to be a suitable value for number of features through prior empirical testing, where more features gave no further improvement, while less features was unable to capture the variation of some classes. The further classes added are: Inline Skate, Watch and Stop Sign for the 5 class, while Camera, Windsor Chair, Revolver, Stegosaurus and Cup are added for the 10 class experiment. These classes are chosen due to low variability in image spatial structure. As the network is only looking for natural spatial structural similarity avoidance of classes which have a large intraclass variance compared to the overall interclass relationship (Zamani and Jamzad, 2017). With this in mind and due to some the additional classes having a smaller number of sequences, the number of training and testing instances was changed to suit, at 20 training and 10 testing. Overall the network was able to achieve classification accuracies of 86% and 75% and IoU values of 76% and 71% for the 5 and 10 classes respectively, results are shown in Table 1. The decrease in overall accuracy with additional classes is to be expected at the features built in the second convolution layer tend to get more similar. This is visually detailed in section 4.3.2 with the Interpretability showing the different features learned in the convolution layers. With this closer similarity of layer-wise features, an example of how the active pruning mechanism is shown in Figure 7, where a number of the features within the second convolution layer have a slower convergence rate while maintaining a high spike activity. This typically suggests the feature is not very discriminative and is an ideal candidate for being reset to learn a new feature. Figure 7, shows the original features just prior to reaching the pruning check point within (a), then indicates which features are chosen to prune with the feature being reset to random initialisation within (b), the finally resulting in new features shown in (c).

Drawing insight from the result, within the 5 class experiment the inter class variance was high. However, once the 10 classes were added this inter and intra class variances seems to overlap. Resulting in many of the classes relying on similar features constructed from circles, with Motorbike, Cup, Camera, Watch, Stop Sign and Face at times producing features are that undistinguishable from one another. It was also noted that as the number of classes increased the difference between average number of features in a kernel per class (that is ones that can be recognised as belonging to a particular class) leads to a higher likelihood that the class with the highest average feature number will be the most active. Within the last experiment with the 10 classes this was prevalent within the Revolver class as it had an average feature count in convolution layer 2 of around 200, while the average for camera was 110. This results in a higher chance that the revolver was classified by mistake ultimately bringing the overall accuracy down.

4.2 Perception, Understanding and Action

This section is split into two parts both further testing the full PUA system, the first continues using the N-Caltech Dataset, however with multiple simultaneous inputs. The second part makes use of recorded data of desk ornament from a hand-held NVS to provide a further example of how the system works within another test environment and how the action part of the system deals with a moving class.

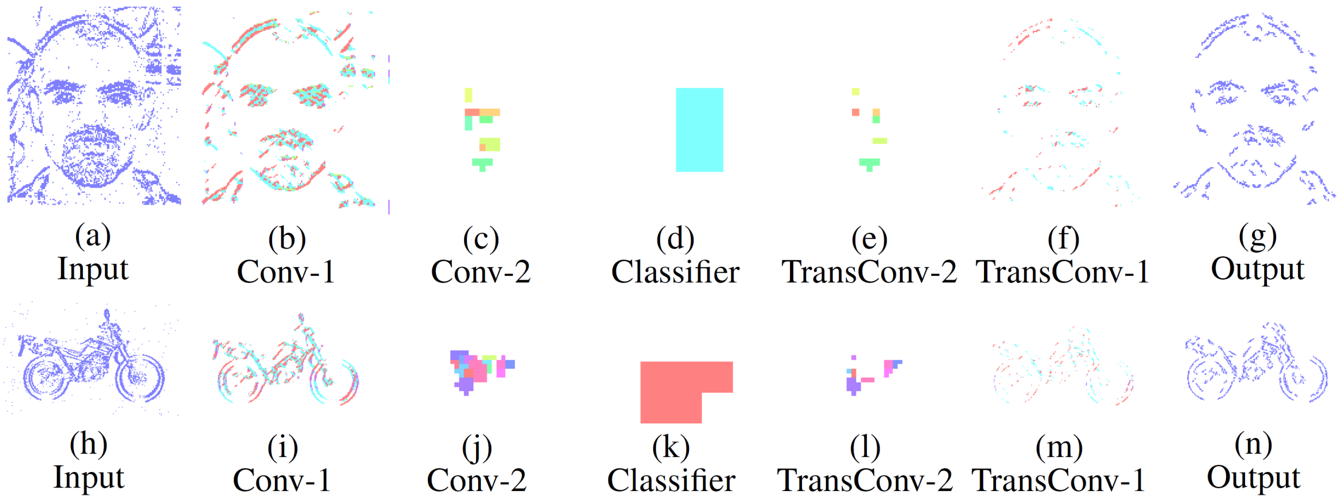


Figure 4. Segmentation performance of the network on an example face (a-g) and motorbike (h-n), highlighting the encoding transition into the latent space used for pseudo classification (a-d, h-k), then retracing of chosen features back to pixel level (d-g, k-n).

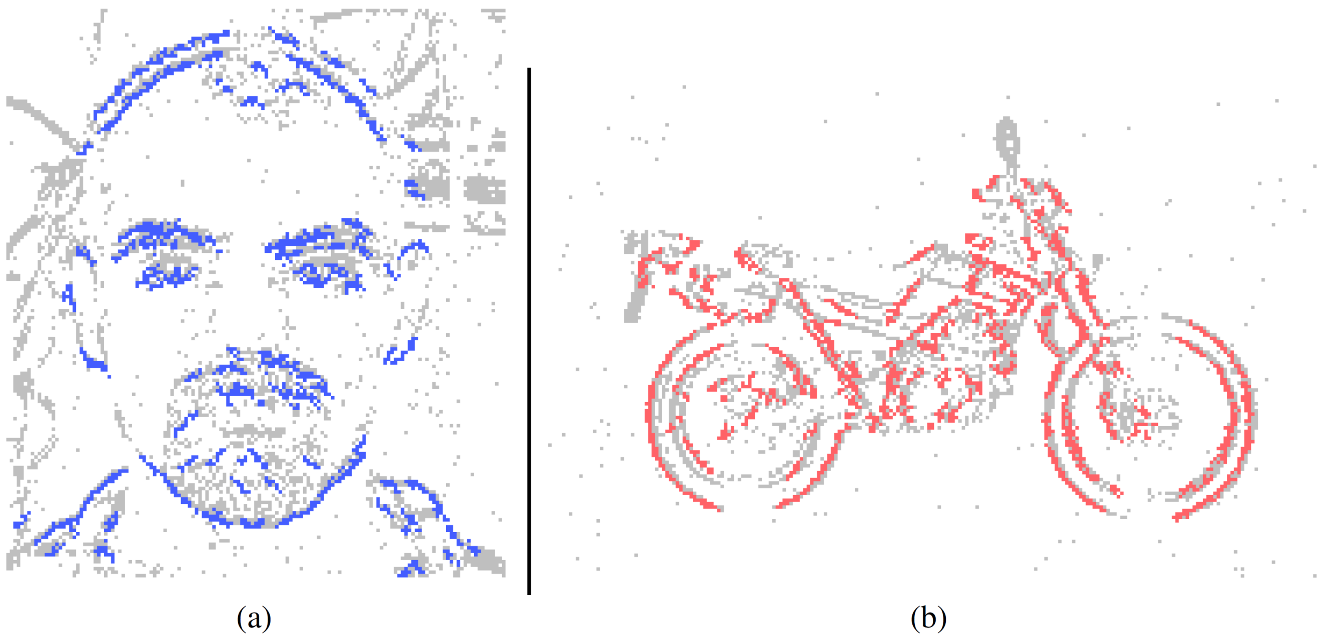


Figure 5. Segmentation overlays for the (a) Face and (b) Motorbike class from the N-CalTech dataset

4.2.1 N-Caltech Mutli-Stream Input

Building upon the results gathered from the successful process in section 4.1, this experiment looks at how the system would deal with multiple input streams. This allows the network to demonstrate the segmentation ability in the face of multiple distractors and spatio-temporal Gaussian noise with an average PSNR of 18dB, an example of the input with and without noise is shown in Figure 8 (b) and (a) respectively. Figure 8 also demonstrates the layout of the new input image, which is based on the Face class subset, but is 3 times the size to make a 3x3 grid where each corner and the centre will host an input stream. Each stream is presented for 300ms (dictated by the recording length in the dataset) then some of the locations

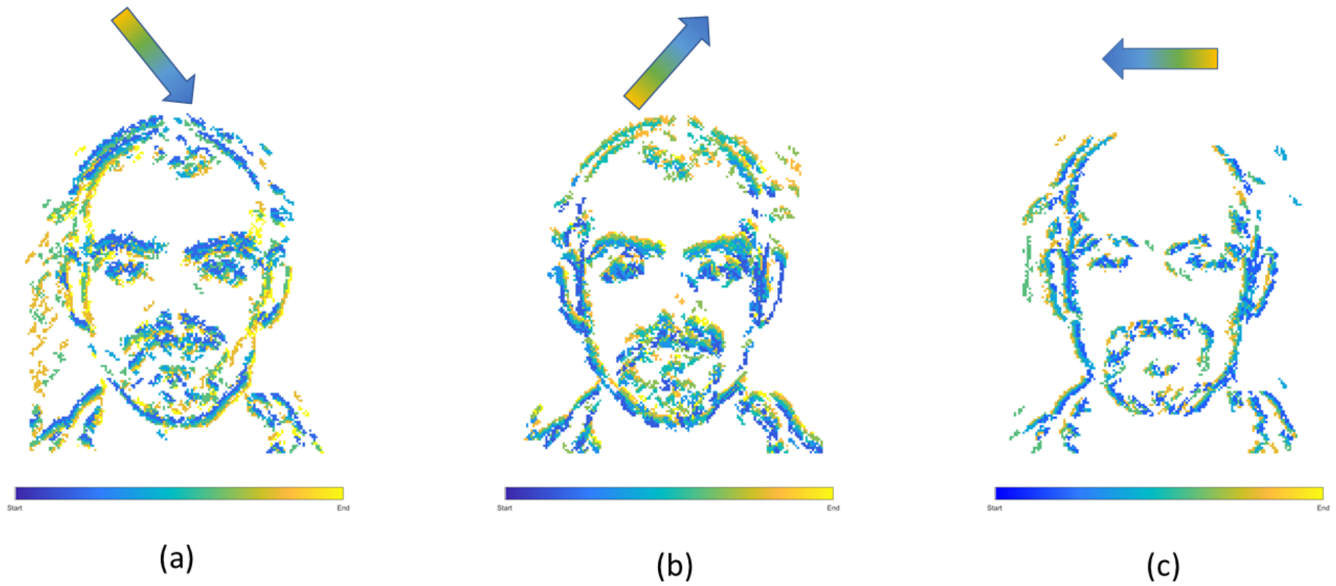


Figure 6. Overlapped Segmentation output over the complete event stream, showing the triangle of movements over the three saccades, (a) first movement, (b) second movement, (c) third movement.

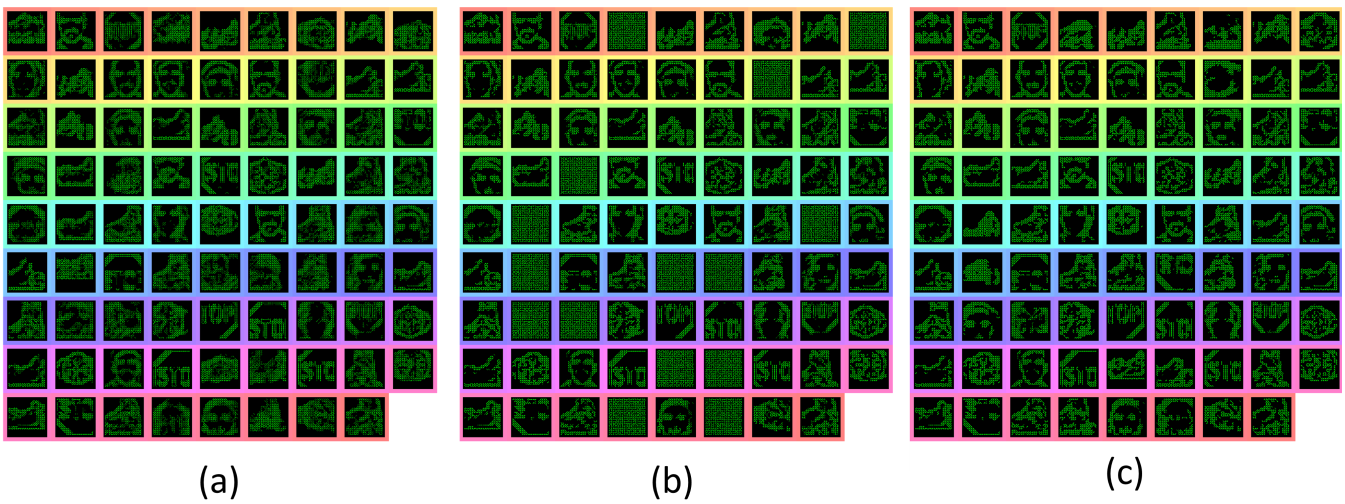


Figure 7. Features from the second convolution layer during training highlighting the pruning process. (a) highlights the features prior to pruning, (b) shows which feature were reset to initial parameters and (c) shows the newly learned features.

are changed and the next stream is played. The input streams illustrated in Figure 8 (a) and (b), consist of 1 face and 2 motorbikes for the known classes and 2 Garfield streams for the unknown, with Fig 8 (b) demonstrating the affect of noise on the input. This gives an opportunity to display the asynchronous layer-wise spike propagation once thresholds have been surpassed, while also offering an insight into how an SNN reduces computational throughout with this thresholding value.

Figure 9 displays both this asynchronous throughput of activity and how the network reduces the numbers of computations, even when presented with noise and distractors, with the time axis showing an accumulation of spikes to ease with visibility. Figure 9 shows that by Conv 1 the added noise is mostly removed as it lacks any real structured shape, but the distractor, Garfield, remains and progresses onto

Conv 2. During this layer though, due to its low saliency with any of the learned features for the classes of Face or Motorbike the distractor is removed from the processing pipeline. This leaves only the two known classes, which then progress onto the Conv3 layer, then through the decoder layers to the output where they are successful segmented. When testing the multi-stream input without any noise the accuracy and IoU value is identical to the single stream instance at 96.8% and 81%. Then with added noise this value sees a slight reduction to 95.1% and 79% for accuracy and IoU, these results are also shown in Table 1. The decreases being attributed to the noisy pixels directly contacting or occurring within the class boundary, as the network has no real way to discern this noise from actual data. This is clearly shown within the segmented output comparisons shown in Figure 10, where the noiseless output (a) and the noisy (b) show considerable difference in their respective segmentations with far more diagonal lines present in the noisy output (b) in comparison to (a). This outcome could have been predicted and will be highlighted in section 4.3 as the first layer of the network has a larger feature representation for the diagonal line when compared to the horizontal and vertical lines, with more pixels allocated to representing the diagonal lines rather than horizontal and vertical, due to the larger variety of edges this feature had to represent. Meaning relatively with the same threshold the diagonal feature is more likely to be activated than the horizontal and vertical.

With the segmentation successfully output the spiking controller now has less spiking activity so should find it easier to be able to track a given class. The tracking starts once the user has made a selection of which class is to become the attention of the network. Experimentally this was tested by selecting the attention after two successful multi class segmentation examples where the stream inputs were repositioned. Figure 11 displays the outputs of the three inputs (a), (b) and (c) with their subsequent paths to segmentation. Figure 11 shows that for inputs (a) and (b) the network is correctly segmenting the input and displaying an output with a highlighted segment displayed in the 3x3 grid. It is only in Fig 11 (c) that the guided attention mechanism is triggered causing the inhibition of the other class in the propagation between layer Conv 3 and T-Conv 3. This feature is highlighted with the red circle showing which neurons are now no longer represented in the subsequent layer and thus no longer computed out to the segmentation, highlighting part of the efficiency in SNN. The last section of the diagram in Figure 11 highlights the attention of the network being drawn to the face located on the bottom left of the grid, which in the spiking controller would result in an output of left and down to ensure the face is located within the central region. The arrow within the Fig 11 (c) also indicated the movement of the track update, which is based off the central region as within the previous two sequences the multiclass attention doesn't give a control output. This attention-based tracking update is delivered within 34ms or 34 input steps, which with the 11ms processing lag with each layer to propagate through the network results in a 31ms delay within the 300ms input stream. This may seem like a considerable amount of time, but as shown in both Figures 2 and 9 due to the way the N-Caltech dataset was recorded, the first 30ms of the recording contains very little information due to the lack of movement with the main concentration of spiking activity during the middle of each of the saccade movements. To test this the first 30ms of events were removed from all the input streams which result in a reduction in track update to 15ms and with the offset of 11ms to progress through the network means a 4-5ms latency to get from input to a control output if the processing could be done in real-time. However, even this latency is mainly from the initial delay in spiking activity within the network first layer, suggesting once running the latency would decrease. This would make it a highly competitive alternative or efficient middle ground between highly precise CNNs and low latency edge detection systems. Furthermore, the total number of average calculations represented by the images seen in Figure 11 is only approx 9% of the total available calculations (equivalent CNN) due to the sparse nature of both the features and the SNN thresholding processing. Approx 10% of capacity is used in the encoding process and approx 5% in the decoding process, which is visualised in both Figures 9 and 11.

4.2.2 Tracking from Handheld NVS

For this section, the SpikeSEG network was retained to be able to identify a panda desk ornament and aims to better highlight the control and tracking aspects of the PUA system. The input stream recorded from DVS346 NVS has the panda start on the far left in the field of view then the camera pans to the left resulting in the panda being on the far right, with an example of the input images shown in Figure 12 (a). The results detail how well the segmentation would work within this example, with the extra complexity of 3D shapes and natural indoor lighting conditions. Overall the results of the 1 second test stream, show that only 60ms (6%) of streaming footage failed to produce a segmentation output. This also occurs at the points where the least amount of movement of the camera happens, the turning points, subsequently producing fewer output spikes. Nevertheless, this results in no actual loss in tracking accuracy as the panda object stayed within the previous segmentations IoU bounding box. Furthermore, the IoU for this test stream was 75% , shown in Table 1, perhaps lower than expected given the high level of accuracy within the classification/segmentation process. This is illustrated in Figure 12 (a) where the middle section of the panda is not well resolved by the sensor, meaning on occasion the segmentation output was only of the top or bottom section. Figure 12 (b), (c) and (d) also show the full system process for the two different control outputs of moving left (d) and right (c), that is when the segmentation area enters the proximity of the spike counter at the edges of the output image. Within Figure 12 (d) there is also an example of how the system overcame a background object that could have affected simpler approaches, as originally the input image had a background object on the right hand side of the image. Due to the feature extraction and segmentation, the background object was unable to influence the controller which without the Understanding-based segmentation would have had spiking activity in both left and right spike counters.

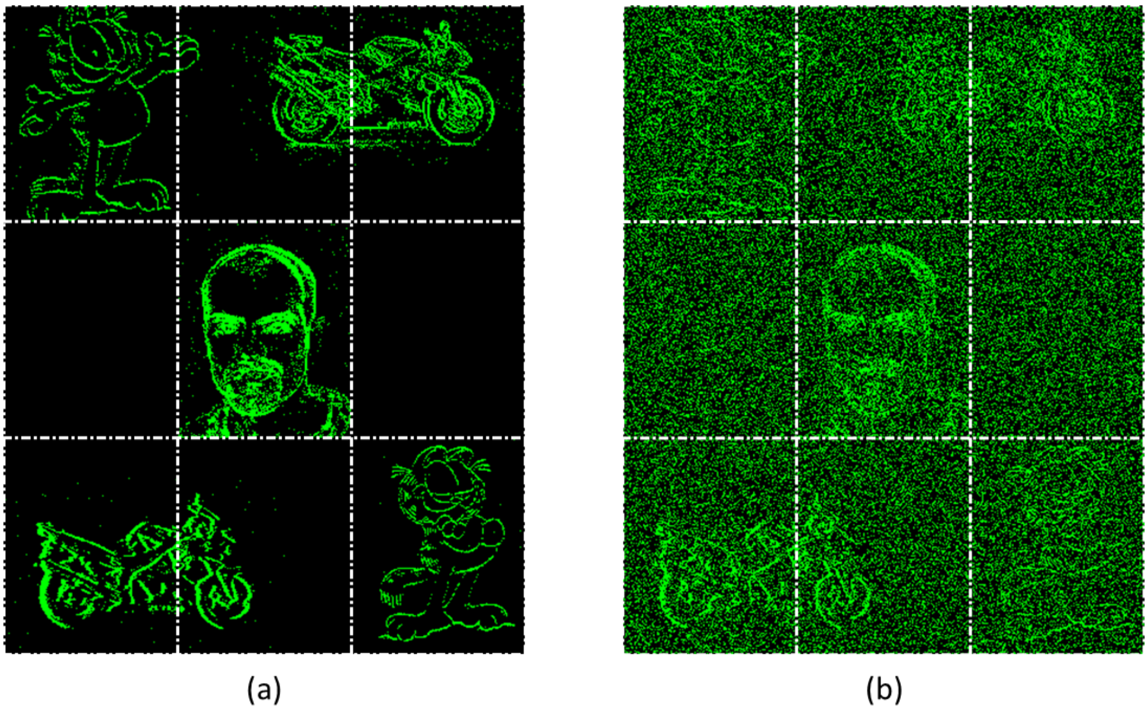


Figure 8. Example of input for the Multi-Stream Input without noise (a) and with noise (b), both with extra gridlines indicating the 3x3 grid which determines the initial location of the inputs.

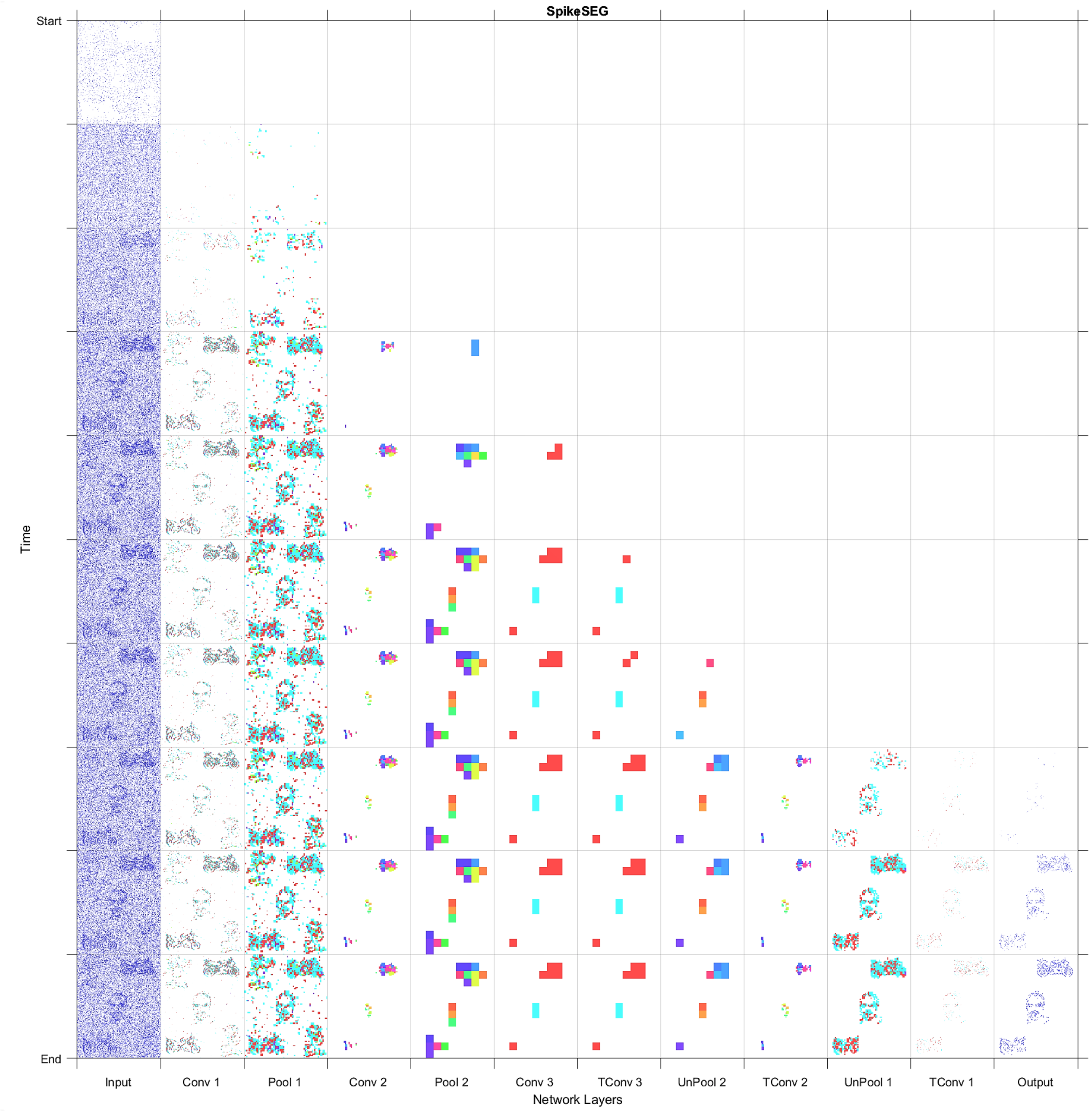


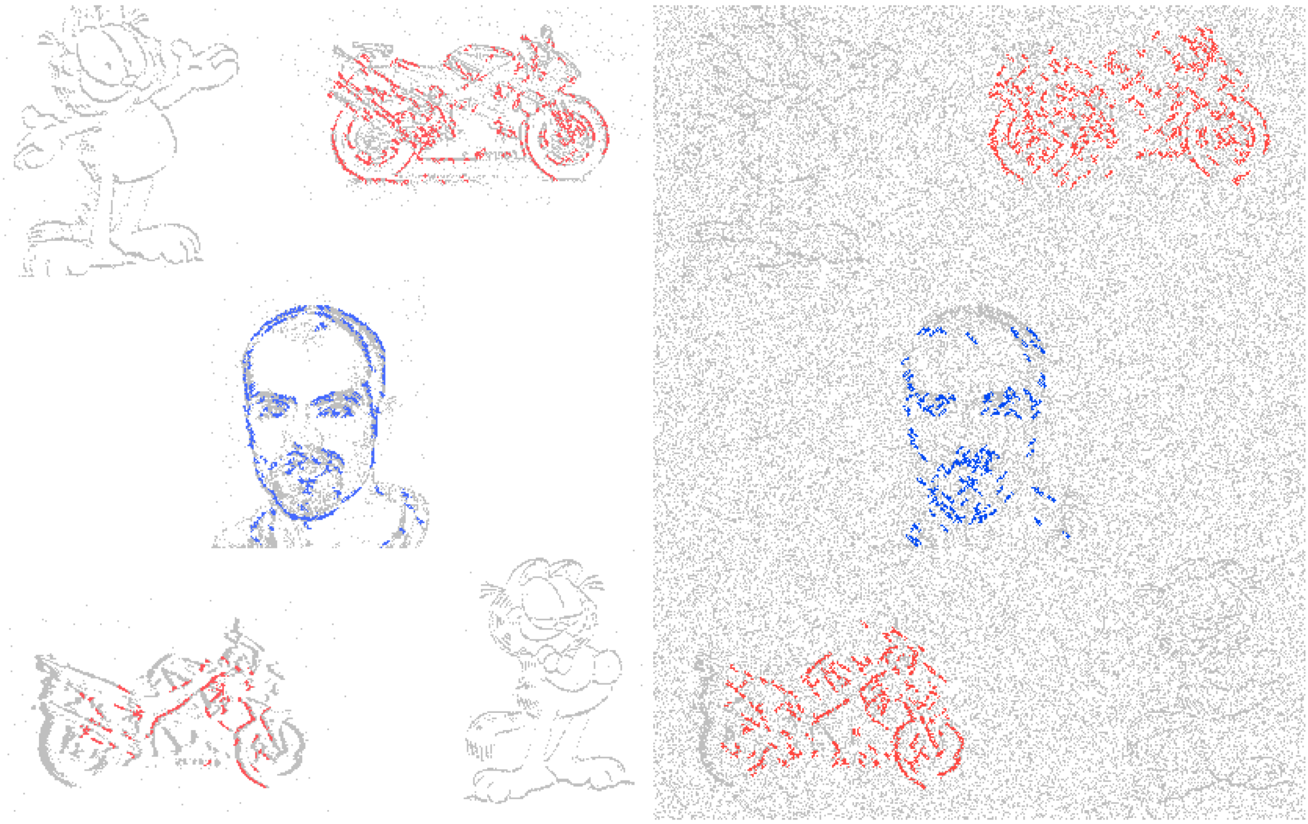
Figure 9. Full Layer-wise spiking activity for the system, showing the progression of spikes through the network encoding then decoding section into the segmentation output

4.3 Robustness and Interpretability

This section highlights two key features of utilising an SNN approach for this framework, the first is system robustness, especially that pertaining to Perception and Understanding (the sensor and processing) and how that affects the Actions of the system. The second feature is that of interpretability something that is not often not associated with CNN type approaches.

4.3.1 Robustness

The added robustness of the PUA approach comes from the Understanding section within the PEAT (Pre-Empt then Adapt Thresholding) mechanism. As mentioned in section 3.3.2, the buffering of input



(a) (b)

Figure 10. Segmentation overlays for the (a) Multi-Stream Input and (b) Multi-Stream Input with noise, including the classes Face, Motorbike and Garfield from the N-CalTech dataset)

spikes allows a spike counter to be implemented, allowing a pre-emptive rather than reactive approach to the thresholding within the network. Permitting synaptic scaling homoeostasis to increase the threshold values on all layers, ensuring noisy or adversarial inputs are mitigated first. Subsequently, if the spike level persists the threshold levels using an intrinsic homoeostasis may be adapted. An example of this system at work is illustrated within Figure 13, with (a) showing a multi-stream input with no noise, then the input is corrupt with noise in (b), (c) and (d) showing the resulting effects of the noise throughout the system with and without the PEAT mechanism active. The PUA framework implements the regime that no output is better than an incorrect output, especially when the input is degraded due to noise or adversarial sensor values. This robustness features is highlighted in the output of Fig 13 (b) which is incorrect and if passed to the controller could cause an undesired response, meanwhile in Fig 13 (c) the PEAT is seen to allow the network to threshold the noise level resulting in no segmentation output. Incidentally, Figure 13 (d) could be the adaptive outcomes of both approaches (b) and (c), it is just intermediate control output suppression that adds an extra level of robustness to the system.

4.3.2 Interpretability

The interpretability of a system is often overlooked when values of accuracy or precision appear to be high. But understanding or gaining some insight into how the system got to an answer could be a valuable advantage for SCNN compared to conventional CNNs. As SCNN trained using STDP happen to produce a sparse feature variation of typical CNN outputs, the SCNN results in features that are more akin of those from contour matching papers (Barranco et al., 2014) while CNNs typically take on features that

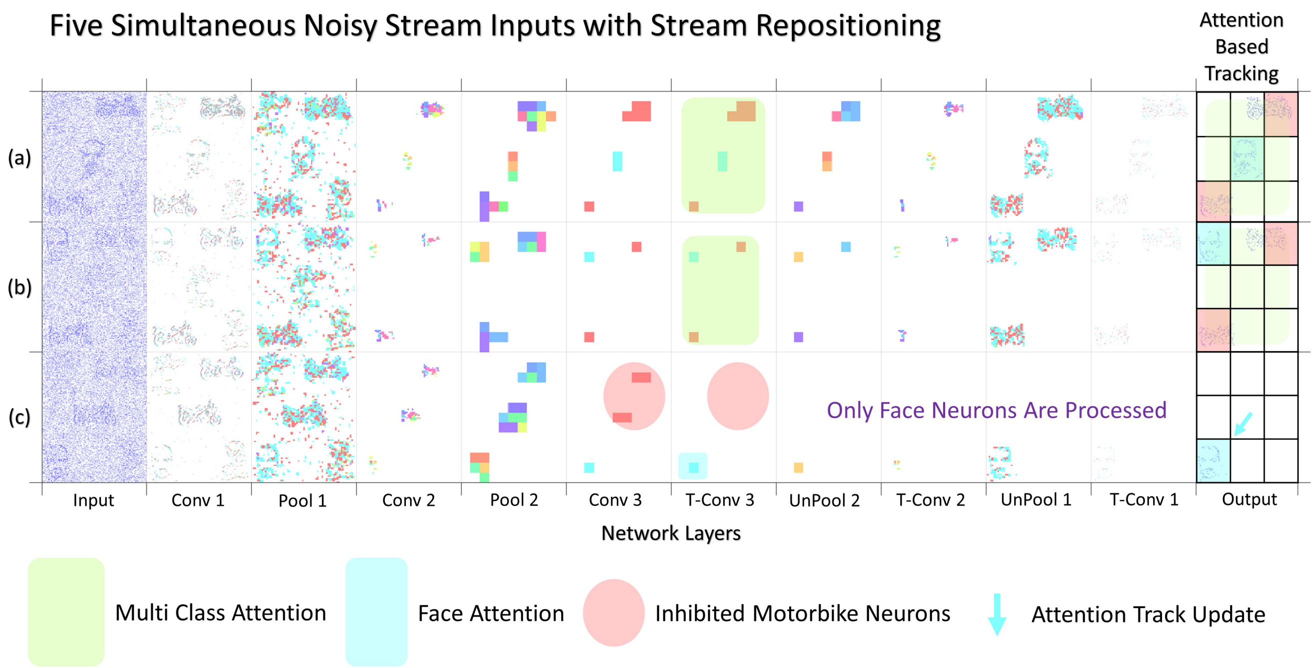


Figure 11. Image showing three separate multi input data streams. (a) and (b) both representing the full system layer-wise computations when no attention is selected, while (c) shows the layer-wise computation after the Face class has been selected as the attention of the network, thus enabling a simplification of the output and activating the action part of the system with a tracking controller update.

resemble textures (Olah et al., 2017). These texture maps are often hard to interpret, although modern approaches have found ways to highlight the most salient parts of an input with reference to these texture maps. Nevertheless it is still often difficult to predict how the system might react to an unknown input. The features that were learned for the testing of the N-Caltech dataset used within this work is shown in Figure 14. Figure14 (a), illustrates the differences between the previous version of the model and the current implementation with PEAT and pruning improving the feature extraction, using the same Conv-1 features representing simple edge detection structure of horizontal, vertical and two diagonal lines. Figure14 (a) then shows the mapping those features onto the weights of the Conv-2 resulting in the features that resemble shapes and objects before the classification stage in Conv-3. It can be seen that half of the 36 features in Conv-2 relate to the Face class and the other half the Motorbike, with these features helping to build up the classification layers with two features either Face or Motorbike. This image helps to explain what the network has learned and how it appears to be looking for contour like shapes to help it distinguish between inputs. Along with this insight into how the network operates, it also allows the user to perhaps understand why it might not always give the correct answers. Similar to how if creating a system using hand-crafted contours features, you would understand the limitation this allows a similar understanding to be had. This could allow manual manipulation of features or manual pruning throughout the training if the user happens to have expert knowledge of the task, bringing neural networks closer to known computer vision-based techniques, which could provide an interesting overlap, especially in the robotics domain.

In order to perceive how the additional classes affects the interpretability of the system Figure 14 (b) and (c) highlights a sub-selection of the features within the 5 and 10 class models. This highlights how the

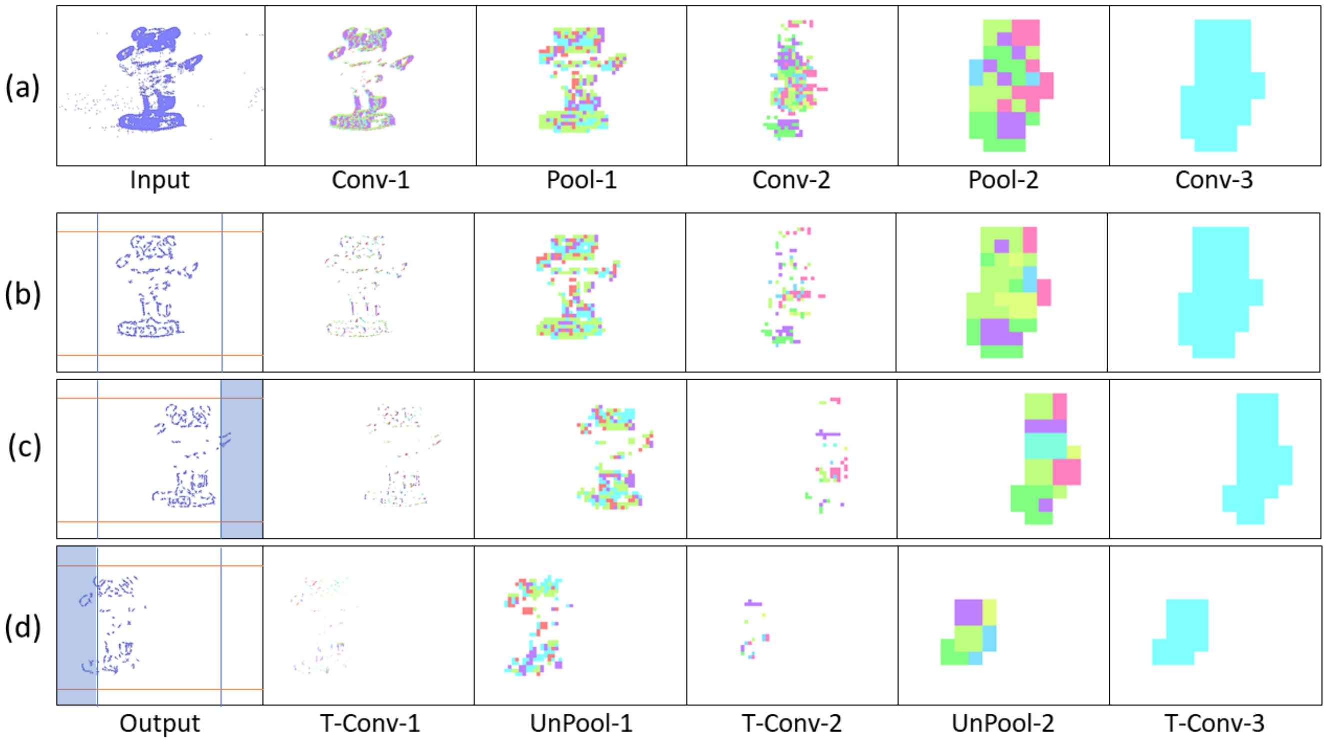


Figure 12. (a) Panda Input Image, (b) Panda reaching rightmost boundary triggering a control action, (b) Panda reaching leftmost boundary triggering a control action.

interpretability is still there for some of the features while others have become more difficult to understand, perhaps due to overlapping features from two classes. Overall, Figure 14 (b) and (c) highlight how reviewing of the features within a Spiking Neural Network can help to gain understanding about parts of the network, with the classification layers features representing each of the 5 and 10 classes. The visualisations help to explain why certain classes might struggle versus others due to similar sub classification features.

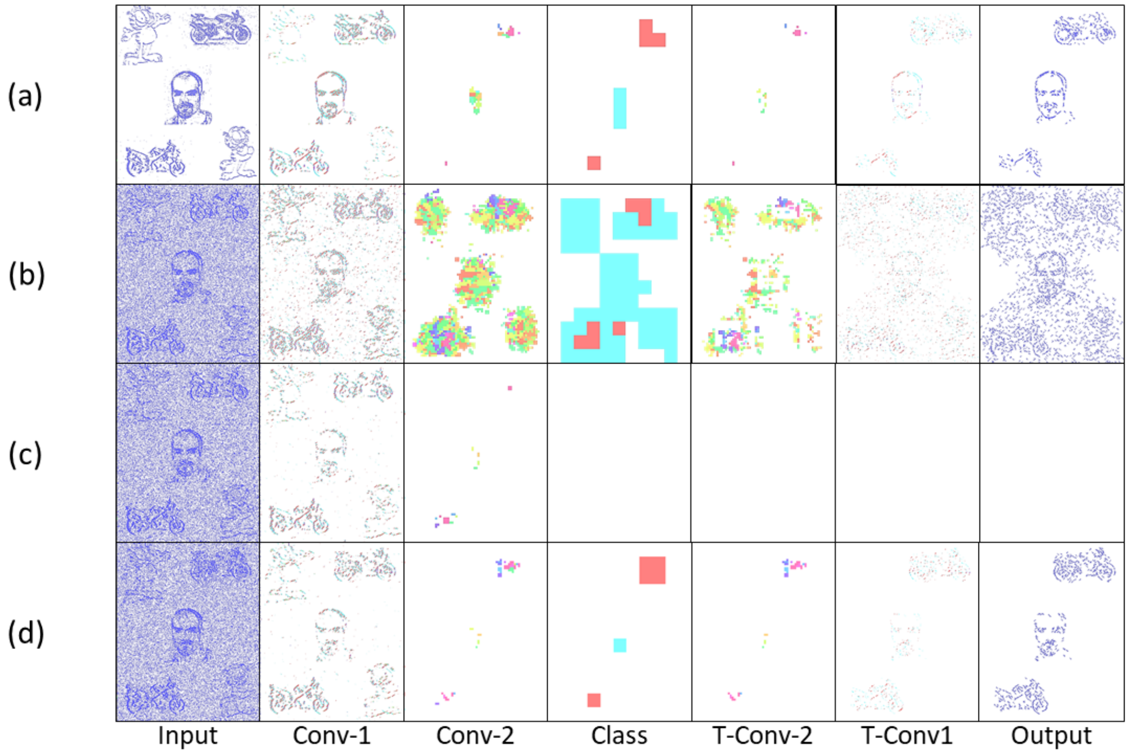


Figure 13. Highlighting the Robust noise suppression with the Pre-Empt then Adapt Thresholding mechanism.

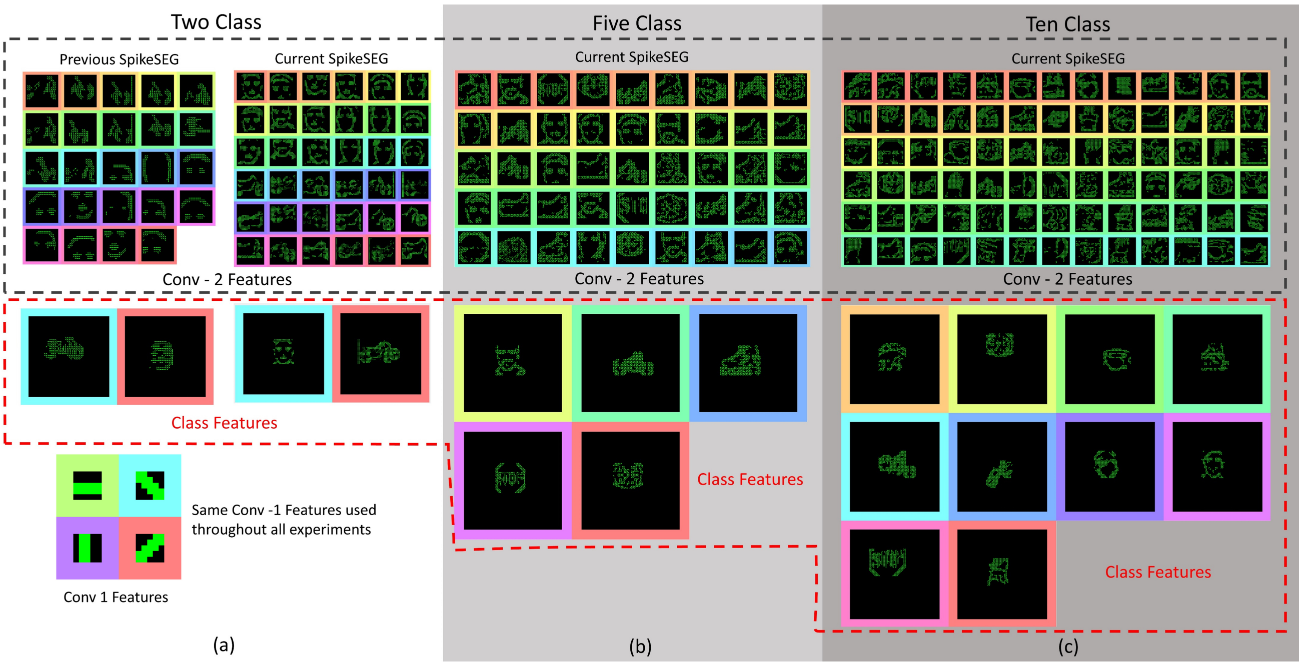


Figure 14. (a) Features map representations of the convolution layers, with colouring to match the latent space representation from the two class experiment, showing prior and current results of the Conv -2 features and Class features. The Figure also shows a selection of features from both the Five Class (b) and Ten Class (c) experiments. Top half showing the Conv -2 features and the bottom showing the Class Features. (a) Classes shown in Class Features are Motorbike-Face then Face-Motorbike for the previous and current results. (b) Classes shown in Class Features order are: Face, Motorbike, In-line Skate, Stop Sign and Watch. (c) Classes shown in Class Features order are: Stegosaurus, Watch, Cup, In-line Skate, Motorbike, Revolver, Camera, Face, Stop Sign and Windsor Chair.

5 DISCUSSION

The understanding method shown in this work details an unsupervised STDP approach. To fully utilise the spiking nature of the processing it is paired with the perception method of spiking input sensor. Together this perception understanding pair can successfully semantically segment up to 10 classes of the N-Caltech dataset. The output of this process is a spiking grid indicating the location of the class within the scene, which can be interpreted by the action system to allow the objects to be followed if attempting to leave the field of view.

The full PUA process is completed in a spiking and fully convolutional manner. This ensures all calculations are either spiking or spike counting. Allowing the network to maintain the temporal and processing advantages, along with the asynchronicity associated with neuromorphic vision sensors, from input to output. However, this method of processing is not without its drawbacks, as there is an overall decrease in accuracy associated with this adding of extra classes. That perhaps indicates the limitation with this unsupervised method in terms of problem scaling. For instances with the 100 classes available within the N-Caltech dataset, the system would only be able to learn the most common features that occur within each class, but only if they present a large enough variance. That is it will only learn common class features as long as they look different enough from the other classes. Which is essentially what can be seen happening with the 5 and 10 class experiments visualised in Figure 14 and Figure 7 (c). Figure 7 (c) highlights that even with a high inter class variance the kernels sometimes learn differentiating features from all other classes, while other times learns features that are an amalgamation of two or more classes. The 5 class experiment displays this most prominently with the Bike and In-Line Skate classes, as there are similarities between the outline shape of both objects.

Nevertheless, this ability to find most common features that express the highest variance from others, is both the limitation and strength of this STDP approach. Limiting in that this approach might not scale to larger datasets, but a strength in that it made the network asynchronous, adaptable, computational sparse and visually interpretable. This highlights that the STDP method used might not be suitable for all problems, but serves as a indication of the benefits if the problem is appropriate. This work demonstrates that STDP alone can be used to find the most common features of a dataset. Which in turn, can be used to successfully perform image classification and semantic segmentation. However, a further learning rule to help focus on more discriminative features such as R-STDP (Izhikevich, 2007; Legenstein et al., 2008; Mozafari et al., 2018) would be a useful extension. This could help in tackling the main issue of inter to intra class variance differentiation. This could allow not only the most common feature to be discovered, but the most common discriminative feature.

6 CONCLUSION

We proposed a new spiking-based system, the Perception Understanding Action Framework, which aims to exploit the low latency and sparse characteristic of the NVS in a fully neuromorphic asynchronous event driven pipeline. Using the understanding gained through the SpikeSEG segmentation, the network is able to detect, classify and segment classes with high accuracy and precision. Then from this understanding, the system makes a more informed decision about what action is to be taken. In this context, the framework was able to show a semantic class tracking ability that combines feature extraction capability of CNNs and low latency and computation throughput of line and corner detection methods. The framework also explores the unique benefits that can be gained through utilising SNNs with interpretability and robustness, with the use of thresholding algorithms and sparse feature extractions. The PUA framework also shows off the unique attention mechanism, emphasizing how simple local inhibition rules when combined with an

Table 1. Results from each of the experimental setup, listing both the accuracy and intersection over union

Dataset	Classification Accuracy (%)	Intersection over Union (%)
N-CalTech (2 Class)	96.8	81
N-CalTech (5 Class)	86	76
N-CalTech (10 Class)	75	71
Multistream N-CalTech	96.8	81
Multistream N-CalTech with Noise	95.1	79
Panda	94	75

encoder decoder structure; this can help reduce the computation overhead of the semantic segmentation process. This research highlighted the series of benefits when utilising a fully neuromorphic approach with a pragmatic engineering and robotics outlook, looking at the biologically inspired mechanisms, features and benefits, then combining them with modern deep learning-based structures.

AUTHOR CONTRIBUTIONS

Author PK is the main author and main contributor to the framework and experimental work. Authors PK, GD and JS contributed to the paper writing. Author GM was employed by the company Leonardo MW ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

ACKNOWLEDGMENTS

This work was support in part by Leonardo MW Ltd, as part of a PhD sponsorship programme. We would like to thank George Matich for discussion, reviewing and his helpful feedback.

REFERENCES

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 2481–2495. doi:10.1109/TPAMI.2016.2644615

Barranco, F., Fermuller, C., and Aloimonos, Y. (2014). Contour Motion Estimation for Asynchronous Event-Driven Cameras. *Proceedings of the IEEE* 102, 1537–1556. doi:10.1109/JPROC.2014.2347207

Bi, G.-q. and Poo, M.-m. (1998). Synaptic Modifications in Cultured Hippocampal Neurons: Dependence on Spike Timing, Synaptic Strength, and Postsynaptic Cell Type. *Journal of Neuroscience* 18, 10464–10472. doi:10.1523/JNEUROSCI.18-24-10464.1998

Bichler, O., Querlioz, D., Thorpe, S. J., Bourgoin, J.-P., and Gamrat, C. (2012). Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity. *Neural Networks* 32, 339–348. doi:10.1016/J.NEUNET.2012.02.022

Bohg, J., Hausman, K., Sankaran, B., Brock, O., Kragic, D., Schaal, S., et al. (2017). Interactive Perception: Leveraging Action in Perception and Perception in Action. *IEEE Transactions on Robotics* 33, 1273–1291. doi:10.1109/TRO.2017.2721939

Brandli, C., Berner, R., Minhao Yang, Shih-Chii Liu, and Delbruck, T. (2014). A 240 x 180 130 dB 3 \mui s Latency Global Shutter Spatiotemporal Vision Sensor. *IEEE Journal of Solid-State Circuits* 49, 2333–2341. doi:10.1109/JSSC.2014.2342715

Cao, Y., Chen, Y., and Khosla, D. (2015). Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision* 113, 54–66

Clady, X., Ieng, S.-H., and Benosman, R. (2015). Asynchronous event-based corner detection and matching. *Neural Networks* 66, 91–106. doi:10.1016/J.NEUNET.2015.02.013

Conradt, J., Cook, M., Berner, R., Lichtsteiner, P., Douglas, R., and Delbruck, T. (2009). A pencil balancing robot using a pair of AER dynamic vision sensors. In *2009 IEEE International Symposium on Circuits and Systems* (IEEE), 781–784. doi:10.1109/ISCAS.2009.5117867

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., et al. (2018). Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro* 38, 82–99. doi:10.1109/MM.2018.112130359

Delbruck, T. and Lang, M. (2013). Robotic goalie with 3 ms reaction time at 4% CPU load using event-based dynamic vision sensor. *Frontiers in Neuroscience* 7, 223. doi:10.3389/fnins.2013.00223

Delbruck, T. and Lichtsteiner, P. (2007). Fast sensory motor control based on event-based hybrid neuromorphic-procedural system. In *2007 IEEE International Symposium on Circuits and Systems* (IEEE), 845–848. doi:10.1109/ISCAS.2007.378038

DeWolf, T., Stewart, T. C., Slotine, J.-J., and Eliasmith, C. (2016). A spiking neural model of adaptive arm control. *Proceedings. Biological sciences* 283. doi:10.1098/rspb.2016.2134

Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., et al. (2012). A large-scale model of the functioning brain. *Science (New York, N.Y.)* 338, 1202–5. doi:10.1126/science.1225266

Falez, P., Tirilly, P., Marius Bilasco, I., Devienne, P., and Boulet, P. (2019). Multi-layered Spiking Neural Network with Target Timestamp Threshold Adaptation and STDP. In *2019 International Joint Conference on Neural Networks (IJCNN)* (IEEE), 1–8. doi:10.1109/IJCNN.2019.8852346

Gehrig, D., Rebecq, H., Gallego, G., and Scaramuzza, D. (2018). Asynchronous, photometric feature tracking using events and frames. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 750–765

Glover, A. and Bartolozzi, C. (2017). Robust visual tracking with a freely-moving event camera. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE), 3769–3776. doi:10.1109/IROS.2017.8206226

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation* 18, 1527–1554. doi:10.1162/neco.2006.18.7.1527

Hunsberger, E. and Eliasmith, C. (2015). Spiking deep networks with lif neurons. *arXiv preprint arXiv:1510.08829*

Izhikevich, E. M. (2007). Solving the Distal Reward Problem through Linkage of STDP and Dopamine Signaling doi:10.1093/cercor/bhl152

Jiang, Z., Bing, Z., Huang, K., and Knoll, A. (2019). Retina-Based Pipe-Like Object Tracking Implemented Through Spiking Neural Network on a Snake Robot. *Frontiers in Neurorobotics* 13, 29. doi:10.3389/fnbot.2019.00029

Kheradpisheh, S. R., Ganjtabesh, M., Thorpe, S. J., and Masquelier, T. (2018). STDP-based spiking deep convolutional neural networks for object recognition. *Neural Networks* 99, 56–67. doi:10.1016/J.NEUNET.2017.12.005

Kim, S., Park, S., Na, B., and Yoon, S. (2019). Spiking-yolo: Spiking neural network for real-time object detection. *arXiv preprint arXiv:1903.06530*

Kirkland, P., Di Caterina, G., Soraghan, J., Andreopoulos, Y., and Matich, G. (2019). Uav detection: a stdp trained deep convolutional spiking neural network retina-neuromorphic approach. In *International Conference on Artificial Neural Networks* (Springer), 724–736

Kirkland, P., Di Caterina, G., Soraghan, J., and Matich, G. (2020). Spikeseg: Spiking segmentation via stdp saliency mapping. In *2020 International Joint Conference on Neural Networks (IJCNN)*. 1–8

Lagorce, X., Meyer, C., Ieng, S.-H., Filliat, D., and Benosman, R. (2015). Asynchronous Event-Based Multikernel Algorithm for High-Speed Visual Features Tracking. *IEEE Transactions on Neural Networks* *and Learning Systems* 26, 1710–1720. doi:10.1109/TNNLS.2014.2352401

Legenstein, R., Pecevski, D., and Maass, W. (2008). A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback. *PLoS Computational Biology* 4, 1000180. doi:10.1371/journal.pcbi.1000180

Levy, S. D. (2020). Robustness Through Simplicity: A Minimalist Gateway to Neurorobotic Flight. *Frontiers in Neurorobotics* 14, 16. doi:10.3389/fnbot.2020.00016

Li, H. and Shi, L. (2019). Robust Event-Based Object Tracking Combining Correlation Filter and CNN Representation. *Frontiers in Neurorobotics* 13, 82. doi:10.3389/fnbot.2019.00082

Li Fei-Fei, Fergus, R., and Perona, P. (2018). Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. In *2004 Conference on Computer Vision and Pattern Recognition Workshop* (IEEE), 178–178. doi:10.1109/CVPR.2004.383

Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 120 dB 15micro s Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE Journal of Solid-State Circuits* 43, 566–576. doi:10.1109/JSSC.2007.914337

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE), 3431–3440. doi:10.1109/CVPR.2015.7298965

Masquelier, T. and Kheradpisheh, S. R. (2018). Optimal Localist and Distributed Coding of Spatiotemporal Spike Patterns Through STDP and Coincidence Detection. *Frontiers in Computational Neuroscience* 12, 74. doi:10.3389/fncom.2018.00074

Masquelier, T. and Thorpe, S. J. (2007). Unsupervised learning of visual features through spike timing dependent plasticity. *PLoS computational biology* 3

Masuta, H., Motoyoshi, T., Sawai, K., Koyanagi, K., and Oshima, T. (2017). Perception and action cycle for cognitive robotics. In *2017 International Symposium on Micro-NanoMechatronics and Human Science (MHS)* (IEEE), 1–7. doi:10.1109/MHS.2017.8305180

Mozafari, M., Kheradpisheh, S. R., Masquelier, T., Nowzari-Dalini, A., and Ganjtabesh, M. (2018). First-spike-based visual categorization using reward-modulated STDP. *IEEE Transactions on Neural Networks and Learning Systems* 29, 6178–6190. doi:10.1109/TNNLS.2018.2826721

Mueggler, E., Bartolozzi, C., and Scaramuzza, D. (2017). Fast event-based corner detection

Nishiwaki, K., Sugihara, T., Kagami, S., Kanehiro, F., Inaba, M., and Inoue, H. (2003). Design and development of research platform for perception-action integration in humanoid robot: H6. In *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)* (IEEE), vol. 3, 1559–1564. doi:10.1109/IROS.2000.895195

O'Connor, P., Neil, D., Liu, S.-C., Delbruck, T., and Pfeiffer, M. (2013). Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in neuroscience* 7, 178. doi:10.3389/fnins.2013.00178

Olah, C., Mordvintsev, A., and Schubert, L. (2017). Feature visualization. *Distill* doi:10.23915/distill.00007. Https://distill.pub/2017/feature-visualization

Orchard, G., Jayawant, A., Cohen, G. K., and Thakor, N. (2015). Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in neuroscience* 9, 437

Panda, P., Srinivasan, G., and Roy, K. (2017). *Convolutional Spike Timing Dependent Plasticity based Feature Learning in Spiking Neural Networks*. Tech. rep.

Park, J., Ha, S., Yu, T., Neftci, E., and Cauwenberghs, G. (2014). A 65k-neuron 73-Mevents/s 22-pJ/event asynchronous micro-pipelined integrate-and-fire array transceiver. In *IEEE 2014 Biomedical Circuits and Systems Conference, BioCAS 2014 - Proceedings* (Institute of Electrical and Electronics Engineers

Inc.), 675–678. doi:10.1109/BioCAS.2014.6981816

Paugam-Moisy, H. and Bohte, S. M. (2012). Computing with Spiking Neuron Networks. In *Handbook of Natural Computing*, eds. G. Rozenberg, T. Back, and J. Kok (Springer-Verlag). 335–376. doi:10.1007/978-3-540-92910-9_10

Paulun, L., Wendt, A., and Kasabov, N. (2018). A Retinotopic Spiking Neural Network System for Accurate Recognition of Moving Objects Using NeuCube and Dynamic Vision Sensors. *Frontiers in Computational Neuroscience* 12, 42. doi:10.3389/fncom.2018.00042

Renner, A., Evanusa, M., and Sandamirskaya, Y. (2019). Event-Based Attention and Tracking on Neuromorphic Hardware. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (IEEE), 1709–1716. doi:10.1109/CVPRW.2019.00220

Seifozzakerini, S., Yau, W.-Y., Zhao, B., and Mao, K. (2016). Event-Based Hough Transform in a Spiking Neural Network for Multiple Line Detection and Tracking Using a Dynamic Vision Sensor. In *Procedings of the British Machine Vision Conference 2016* (British Machine Vision Association), 94.1–94.12. doi:10.5244/C.30.94

Sengupta, A., Ye, Y., Wang, R., Liu, C., and Roy, K. (2019). Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in neuroscience* 13

Simonyan, K., Vedaldi, A., and Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*

Stromatias, E., Soto, M., Serrano-Gotarredona, T., and Linares-Barranco, B. (2017). An Event-Driven Classifier for Spiking Neural Networks Fed with Synthetic or Dynamic Vision Sensor Data. *Frontiers in Neuroscience* 11, 350. doi:10.3389/fnins.2017.00350

Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., and Maida, A. (2019). Deep learning in spiking neural networks. *Neural Networks* 111, 47–63. doi:10.1016/J.NEUNET.2018.12.002

Vasco, V., Glover, A., and Bartolozzi, C. (2016). Fast event-based Harris corner detection exploiting the advantages of event-driven cameras. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE), 4144–4149. doi:10.1109/IROS.2016.7759610

Wiesmann, G., Schraml, S., Litzenberger, M., Belbachir, A. N., Hofstatter, M., and Bartolozzi, C. (2012). Event-driven embodied system for feature extraction and object recognition in robotic applications. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* (IEEE), 76–82. doi:10.1109/CVPRW.2012.6238898

Xie, M. (2003). *Fundamentals of Robotics*, vol. 54 of *Series in Machine Perception and Artificial Intelligence* (WORLD SCIENTIFIC). doi:10.1142/5230

Zamani, F. and Jamzad, M. (2017). A feature fusion based localized multiple kernel learning system for real world image classification. *EURASIP Journal on Image and Video Processing* 2017, 78. doi:10.1186/s13640-017-0225-y

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision* (Springer), 818–833

Zitnick, C. L. and Dollár, P. (2014). Edge Boxes: Locating Object Proposals from Edges (Springer, Cham). 391–405. doi:10.1007/978-3-319-10602-1_26