Unsupervised Spiking Instance Segmentation on Event Data using STDP Features

Paul Kirkland, Davide Manna, Alex Vicente, and Gaetano Di Caterina, *Member, IEEE*

Abstract—Spiking Neural Networks (SNN) and the field of Neuromorphic Engineering has brought about a paradigm shift in how to approach Machine Learning (ML) and Computer Vision (CV) problem. This paradigm shift comes from the adaption of event-based sensing and processing. An event-based vision sensor allows for sparse and asynchronous events to be produced that are dynamically related to the scene. Allowing not only the spatial information but a high-fidelity of temporal information to be captured. Meanwhile avoiding the extra overhead and redundancy of conventional high frame rate approaches. However, with this change in paradigm, many techniques from traditional CV and ML are not applicable to these event-based spatial-temporal visual streams. As such a limited number of recognition, detection and segmentation approaches exist. In this paper, we present a novel approach that can perform instance segmentation using just the weights of a Spike Time Dependent Plasticity trained Spiking Convolutional Neural Network that was trained for object recognition. This exploits the spatial and temporal aspects of the SpikeSEG network's internal feature representations adding this new discriminative capability. We highlight the new capability by successfully transforming a single class unsupervised network for face detection into a multi-person face recognition and instance segmentation network.

Index Terms—Neuromorphic Engineering, Neuromorphic Algorithms, SNN, STDP, Computer Vision, Unsuperivsed Learning, Instance Segmenation

1 Introduction

HUMAN vision has the innate ability to detect, localise, and differentiate objects of interest, even in the face of multiple background and foreground distractors. Meanwhile, in Computer Vision (CV), this ability known as instance segmentation is a complex and challenging task. Instance segmentation is the detecting and pixel-wise delineation of each distinct object of interest appearing in an image. In essence instance segmentation is the hybrid of two key computer vision tasks, object detection and semantic segmentation. Object detection is the process of detecting instances of objects belonging to a certain class, while also identifying their spatial location typically with a bounding box. While semantic segmentation is the task of clustering parts of an image together that belong to the same object class, resulting in a much more detailed pixel-wise localisation.

The main component of an object detector for segmentation is a good internal feature representation [1], [2]. Traditionally, considerable effort was put into designing handcrafted local descriptors: SIFT [3], [4], HOG [5] and SURF [6]. These low-level features are then utilised to create high-level image representations through Bag of Words [7] and Fisher Vectors [8]. Which in turn, are then used for classification. The downside to these feature representation methods is that often the internal feature representation was based on the level of domain expertise. Lately, with the advent of Deep Learning (DL) [9], beyond state-of-the-art performance [10] was now reachable without any domain expertise and handcrafted features for a multitude of tasks: image recognition (AlexNet and ResNet) [11], [12], object detection (Faster R-CNN and RFCN) [13], [14], object tracking (FCNT and MD Net) [15], [16] and semantic and instance segmentation (FCN, SegNet, U-Net and Mask R-CNN) [17], [18], [19], [20]. This improvement is mainly due to the combination of feature extraction and classification now being governed by supervised techniques using backpropagation. Consequently, this shifted the problem of internal feature representation to the development of bigger and better neural networks, where the goal of better segmentation accuracy often transcends other factors, such as segmentation efficiency. This leads to a stark increase in the real-time computational and memory overhead of the system [21], [22]. Not to mention the burden of large volumes of labelled pixel-wise segmentation images for training. This is especially true for deployable systems, where the tight constraints on computation are most prevalent [23], while also requiring an accurate, real-time application. This often leads to an impasse in terms of efficiency and accuracy, with the traditional methods being more efficiency-focused, while the DL methods being more accuracy-focused. Coupled with these, other desirable features of deployable systems are adaptability and continual learning [24], allowing the system to better deal with changes in the environment and unprecedented situations. However within the current literature of SNNs, a variety of compromises are still being made in terms of a full neuromorphic system for segmentation. [25], [26] both utilise traditional images and SNNs trained using supervised backpropagation techniques. [27] makes use of traditional images and neuromorphic processing to perform segmentation in noisy images. Lastly, [28] present SpikeMS, which is able to segment motion from an event

- *Paul Kirkland, Davide Manna, Alex Vicente, and Gaetano di Caterina are with the Neuromorphic Sensor Signal Processing Lab, Centre for Image and Signal Processing, Electrical and Electronic Engineering, University of Strathclyde, Glasgow, United Kingdom.*
  E-mail: paul.kirkland@strath.ac.uk

Manuscript received November 1st, 2021

This is a peer-reviewed, accepted author manuscript of the following article: Kirkland, P., Manna, D., Vicente Sola, A., & Di Caterina, G. (2022). Unsupervised spiking instance segmentation on event data using STDP features. IEEE Transactions on Computers. https://doi.org/10.1109/TC.2022.3191968

stream, though still makes use of a backpropagation method for learning.

In this work, we demonstrate that from a simple unsupervised Spike Time Dependent Plasticity (STDP) [29] trained objection recognition Spiking Convolutional Neural Network (SCNN) [30], [31], our method can extend this to object detection and instance segmentation. Additionally, this occurs without any further learning, so in essence, can be strapped onto any STDP-trained network. STDP has the ability to produce a sparse and temporally variant latent space representation of the input data. These features can then capture the most salient spatial features within its SCNN kernels (weight), where the temporal element captures the order of the feature's prominence. The aim is to maintain all the useful features of event sensing; asynchronous, event-based, sparse, and high temporal resolution. Then exploit these using a processing method designed to exploit the sensing features and maintain the neuromorphic processing benefits; event-based, asynchronous, adaptation, sparsity, high temporal fidelity and a low computational throughput. These features could help to enable embedded systems smaller than currently possible that are able to perform on system real-time processing.

Our proposed method, called Hierarchical Unravelling of Linked Kernels and Similarity Matching through Active Spike Hashing (HULK SMASH), presents a technique of utilising the internal featural-temporal representation to gain this instance segmentation ability, that is using the changing internal feature representation over time to further discriminate (i.e. sub-cluster) the intra-class representations. HULK SMASH utilises the inherent saliency of the STDP spatial-temporal features captured by the original network for each layer. Meaning it no longer just classifies from the upmost hierarchical layer of the SCNN. Instead utilising the temporal occurrence of internal feature representations to take an unsupervised network trained to recognise one class, faces, and extend it to identify and localise individuals. This idea builds on the efficient use of low-level features of traditional CV methods while using networks and techniques from DL to build better hierarchical features and achieve better accuracy. SNNs provide the ideal platform for this hybrid approach, while further extending the feature set of the implemented network.

The HULK SMASH algorithm is not only able to exploit all the useful features of Neuromorphic Engineering, but adds a further level of interpretability. This is in part due to the use of sparsity of the visual features making them easy to realise, while also exploiting the saliency mapping of the SCNN to help visualise which pixels and features played a part in the classification process. This allows the overall interpretability of the network to increase, with the decisions made by the network to become viewable and assessable, so removing the black box stigma of neural networks. This work makes use of previous research that showed how an SCNN trained with STDP could first perform image classification [32], and recently how it can be transformed into an encoder-decoder network, allowing semantic segmentation to be realised [33], [34]. However, this previous research could not distinguish instances within these semantic segmentations. Therefore, the main contributions of this paper are:

- The instance-wise mapping of semantic labels through the new decoder, HULK. This allows the individual spiking instances of the deepest convolutional layer (the pseudo classification layer) of a fully convolution encoder-decoder to be tracked back to the pixel domain. This tracking illustrates that each semantic label is often built up from multiple spiking instances from within the pseudo classification layer. This results in a meta analysis of the overall activity within the SpikeSEG network.
- The semantic instances from HULK are then passed onto SMASH, which looks at SpikeSEG's internal featural-temporal representation and uses this to compare instances. Allowing a metric in which to form an evaluation of similarity between instances. This is then coupled with a locality checking algorithm which then reports a SMASH score for each instance. This allows similar instances can combine to form larger representations for objects. Failing that, creating a new object for any dissimilar instances.

The formulation of the algorithms HULK and SMASH is described in Section 2, illustrating how to extrapolate the internal feature representation from a network. Section 3 details the experimental results used to gauge the ability of SpikeSEG to now perform object detection and instance segmentation. This is also extended into situations where partial and full occlusions of objects happen, with the final experiment showing how HULK SMASH performs on occlusion recovery. Section 4 presents details on the interpretability of the HULK SMASH. Discussing how visualising the internal feature representation of the neural network can help with explainability. This leads to how the original object recognition network was extended for object detection and instance segmentation task through the means of saliency mapping. Section 5 then provides the conclusion of the paper.

2 Temporal Spike Matching

The proposed temporal spike matching algorithm HULK SMASH makes use of an image classification SNN trained using STDP, similar to that seen in [30], [31], [32]. Prior developments have shown that extending an image classification SNN to a semantic segmentation network was possible [33], [34]. These two prior works also utilised the N-Caltech dataset [35] making use of the event driven nature of the input sequences. This work presents how semantic segmentation can also be further extended into an efficient method for object detection and instance segmentation, where instead of using a regression process to indicate which classification instance are connected, as seen with DL models [17], [18], [19], [20], a proximity and temporal-featural (time- and feature-based) similarity metric is used. This allows the temporal occurrence of the internal feature representation, brought about by the temporal dynamic of the input event stream, to further discriminate intra-class instances. As illustrated in Fig. 1, this method of temporal spike matching for instance segmentation can be broken down into two main parts, the *Intra-* and *Inter-Sequence Processing*: in other words, what is happening internally and externally throughout the process. The intra-sequence

is where most of the processing happens, while the inter-sequence allows the processing to link to other instance of the process running, e.g. to compare object instance to see if they are the same object. This can be seen in a block diagram of the proposed method in Fig. 1, within the green and red dot-dashed boxes.

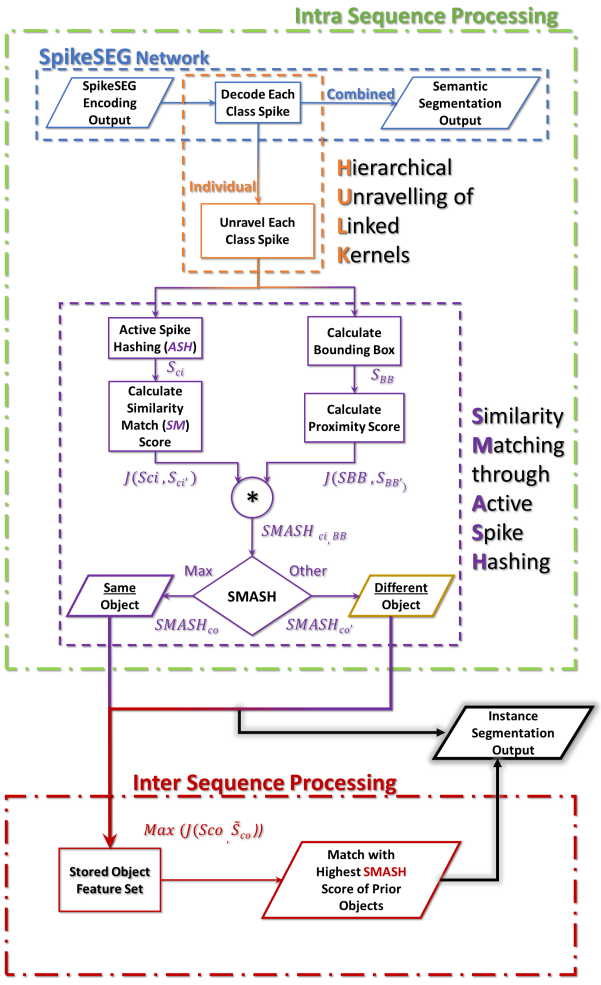

Fig. 1. Block Diagram of the HULK SMASH system and where it intersects with the SpikeSEG Network

Fig. 1 highlights that the *intra-sequence Process* starts at the intersection of the encoder-decoder network from SpikeSEG [33]. The first step of the process is to individually process each of the spiking instances from the classification layer of the SpikeSEG network. This is in contrast to the original network, which just grouped all instances based on class. The process of decoding each instance individually is what is referred to as the HULK process. Once each classification spike has been decoded back into the original pixel space, this can then be feed to the SMASH process. This in turn calculates a bounding box of each instance S_{BB} in the pixel space, by taking the max and min value in the x and y coordinates. This bounding box is then it used as a *Proximity Score* check with every other instance's bounding box $S_{BB'}$ within the *intra-sequence*. This *Proximity Score* is calculated with the Jaccard Index $J(S_{BB}, S_{BB'})$ [36]. Meanwhile, the other parallel process performs the *Active Spike Hashing (ASH)* part of the SMASH process, which stores 2D featural-temporal data S_{ci} from the 4D spatial-featural-temporal decoding class instance processing $(X, Y, FeatureMapandTime)$. This then passes onto the *Similarity Matching (SM)* phase, where a similarity score is given to the featural-temporal data compared with every other instance $S_{ci'}$ within the *intra-sequence*, again with the Jaccard index $J(S_{ci}, S_{ci'})$. The combination of both the Similarity and Proximity Score gives the SMASH score $SMASH_{ci,BB}$. This indicates if the S_{ci} and S_{BB} of that particular class instance match any other $S_{ci'}$ and $S_{BB'}$, resulting in an outcome of whether the instance is part of the same *Object* $SMASH_{co}$ or a different one $SMASH_{co'}$. This intra-sequence process is run in parallel once all the instances have their ASH and bounding box process complete. This is run for every input sequence of data. Which in this case is 10ms of NVS N-Caltech101 [35] event data, which is internally treated as an asynchronous stream based on the timestamps of the events. This allows both object detection and instance segmentation to be performed on this spiking input sequence.

2.1 The SpikeSEG Network

As described the HULK SMASH process makes use of a STPD trained semantic segmentation network SpikeSEG, first seen in [33] then improved and expanded within [34]. An illustration of the fully convolutional spiking encoder-decoder network is shown within Fig. 2. Where the encoder side of SpikeSEG was trained using STDP on the N-Caltech Dataset [35]. The decoder side of SpikeSEG is then attached to the encoder with the same trained weights as the encoder and decoder layers are mirrored. Utilising max pooling indices and a temporal delay signal (to maintain temporal continuity of the spikes) the classification spikes are then transformed via the transposed convolutions and unpooling layers back into the pixel domain. This process results in a saliency mapped segmentation in the output of SpikeSEG. The HULK process then intersects this SpikeSEG network at the classification node, allowing the instance-wise, rather than class-wise segmentation process to commence. The STDP nature of the features extracted by the encoder plays a key role in how the HULK and SMASH processes operate. HULK makes use of the sparse nature of the features extracted through the use of a winner take all STDP implementation and how combined with an adaptive threshold produce sparse internal spiking activity [34]. Without the sparsity in the classification layer, the unravelling process would be overly complex and computationally expensive. Another inherent exploit of the STDP features is their ability to encode both time and saliency. The temporal and sequential nature of the spiking information is retained with the SpikeSEG network. Furthermore, the saliency of features when matched in the input data is encoded into earlier spiking occurrence. This means the order in which feature spikes occur can determine the distinctiveness of features present in the scene. It is this principle that the SMASH process exploits within the similarity matching regime, hypothesising that this feature occurrence helps in differentiating instances from one another. Enhancing the rather simplistic spatial features of STDP with this extra dimensionality of temporal/sequential information.

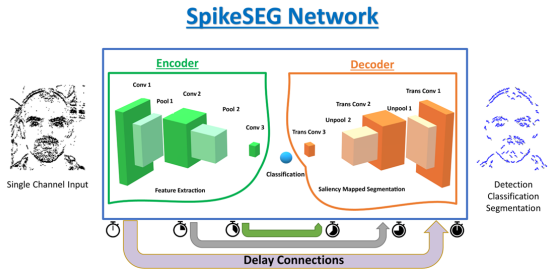

Fig. 2. Illustration of the SpikeSEG network

2.2 Hierarchical Unravelling of Linked Kernels

The Hierarchical Unravelling of Linked Kernels (HULK) process permits spiking activity from the classification convolution layer, to be tracked as it propagates through the decoding layers of HULK. It in essence works the same as if you passed each classification spike into a separate SpikeSEG decoding network, so that only that spiking instance in the classification layer is mapped to the pixel domain, showing directly the features and pixels that cause the classification spike. The Hierarchical Unravelling is referring to the process of disentangling each spike in the classification layer through each of the decoding layers back into the pixel space. Where this process is also seen as the linked kernels, as we are linking the spike in the classification layer back to the pixel space through a series of kernels (weight matrices). This tracked propagation of each instance allows a record of the spiking activity of each subsequent layer's spatial (x, y), featural (m) and temporal (t) spiking activity. This process is illustrated within Fig. 3, with a typical semantically decoded sequence shown entitled 'Accumulated Decoded Spiking Activity' from the SpikeSEG network. Underneath the instance representation of HULK, with the individual class spike breakdown shown within Instance A through D. Each instance, in this case, is representing a single spiking pixel from the 4 shown spikes in the class layer. Whereas in semantic segmentation, all the instances belonging to one class are treated as the same entity and decoded. Fig. 3 highlights how the class instance is broken down into its individual instances, where each instance often provides enough information to recreate the face in the output pixel space. Although, it is clear that some of the instances favour certain features over others. It is also apparent that there is much repetition and accumulation of features even with this sparse spiking domain. It is through this process of unravelling the classification spiking activity that permits the ensuing similarity matching process.

2.3 Similarity Matching through Active Spike Hashing

The Active Spike Hashing (ASH) is the process of taking the recorded spiking activity and implementing an efficient and effective way to store the sparse 4 dimensional spatial (x, y), featural (m) and temporal (t) values. This is done by realising that the convolution structure of the SCNN SpikeSEG is already dealing with the translational invariance and spatial dimension, together with the bounding box proximity score. It can also be noted that position within the scene is not a useful evaluation metric of the similarity between two

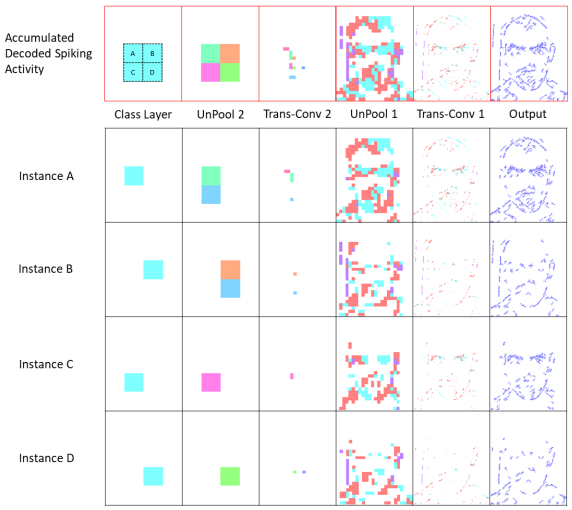

Fig. 3. Illustration of the individual class spikes being unravelled, indicating activity caused by these class spikes

objects. The ASH process then results in a 2D featural-temporal hashing of the spiking activity. This essentially works as a spike train with each feature neuron activity over time being mapped out. Where the featural data then has the same total number of features as the network, which in this case, is 41. This value comes from the use of a network with a similar structure as the Face-Motorbike Network used Kirkland et al. [34], except with only one classification layer. The other 40 features come from the 36 from Trans-Conv2 and 4 from Trans-Conv1.

A further memory reduction to the ASH process is permitted by storing the values as binary terms. As the number of spikes recorded within each map per timestep is more a measure of the total spiking activity, rather than the featural-temporal characteristics we evaluate with the similarity measure.

An illustration of the ASH process where the spiking activity is assigned its feature map and time-stamp index (f, t) can be seen in Fig. 4, along with the associated spike train output of two different faces.. This builds upon the HULK process previously seen in Fig. 3 with the indices of the spiking activity now visible in Fig. 4 for each convolutional layer. Each active neuron is assigned an f and t index on a class-instance basis. Allowing a 2D matrix of each instance to be realised where from Fig. 4, Instance A would have 1s in the first column $(time)$ forth row $(featuremap)$ for the classification layer activity. Meanwhile, the green coloured features of Instance A's Trans-Conv 2, are stored with row 19 and columns 3 and 5. The ASH process typically reduces the size of the tensor by 98%, reducing the memory overhead considerably. The result of the ASH process leaves the internal feature representation similar to that of a conventional spike train as seen on the right of Fig. 4. This spike train shows two different face instances, where the neurons in the first layer (Neurons 1-4) have a similar featural-temporal response, while the deeper layers have a distinctive feature pattern to one another.

Once the spiking activity is hashed, it is ready for

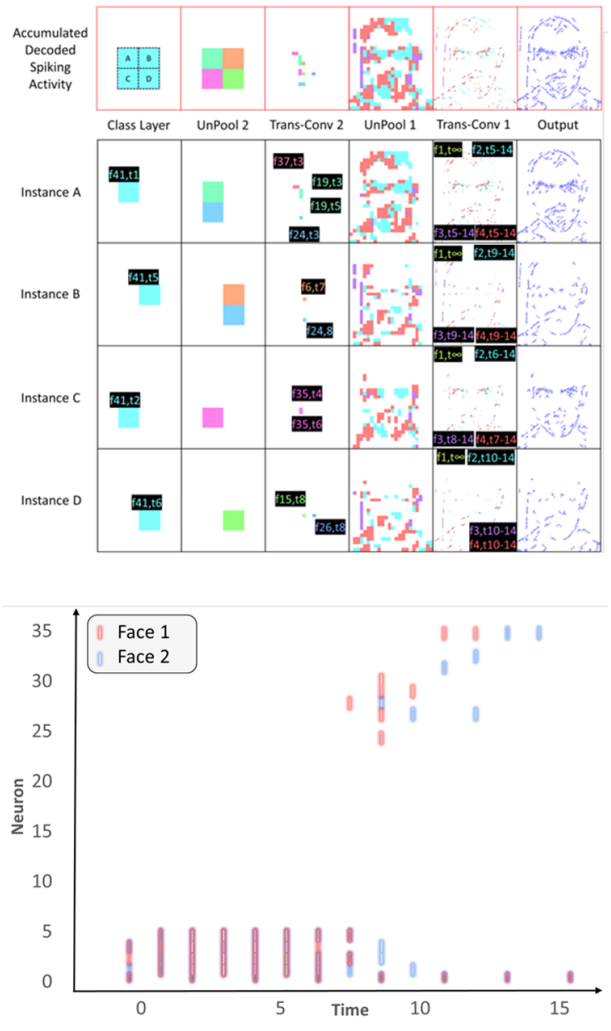

Fig. 4. Further explanation of the Active Spike Hashing mechanism from the HULK process, feature map and time given to each active neurons

the Similarity Matching section of SMASH. The Jaccard similarity coefficient was once again utilised, within the intersection over union used to calculate the segmentation bounding box overlap. However, within this instance, the measure is used to see how much similarity there is from each instance in the intra-sequence process, comparing the featural-temporal similarity of each instance's spiking activity, with respect to the overall amount of spiking activity [36].

$$J(S_{Ci}, S_{Ci'}) = \frac{|S_{Ci} \cap S_{Ci'}|}{|S_{Ci} \cup S_{Ci'}|} \tag{1}$$

where S_{Ci} is the spiking activity of that class instance, and $S_{Ci'}$ represents the spiking activity of any other class instance for that sequence. However, due to the ASH process storing binary values of the activities, this equation can be simplified down to a logical calculation performed with only ORs and ANDs. This allows the quick comparison of the number of spikes that feature in both the current instance *and* the comparison, divided by the number of spikes that feature in the current instance *or* the comparison.

$$J(S_{Ci}, S_{Ci'}) = \frac{S_{Ci} \wedge S_{Ci'}}{S_{Ci} \vee S_{Ci'}} \tag{2}$$

To complete the intra-sequence of the SMASH process, the bounding box IoU score must be calculated for each class instance bounding box Ci_{BB} against the other instances bounding boxes $Ci_{BB'}$, instead of for each class in total

$$J(Ci_{BB}, Ci_{BB'}) = \frac{|Ci_{BB} \cap Ci_{BB'}|}{|Ci_{BB} \cup Ci_{BB'}|} \tag{3}$$

Multiplication of the similarity score, with the IoU, results in the novel proposed SMASH score for each instance

$$\text{SMASH}(Ci, Ci') = J(S_{Ci}, S_{Ci'}) \times J(Ci_{BB}, Ci_{BB'}) \tag{4}$$

Once the SMASH score is calculated for each instance, the maximum of each instance is assigned to a class object Co

$$S_{Co} = \arg\max_{Ci}(\text{SMASH}(Ci, Ci')) \tag{5}$$

This maximum SMASH score is based on the pairing Ci, Ci', where each object S_{Co} is compiled from any overlapping parings e.g. for 5 instances the argmax presents the following sets of Ci, Ci' pairs, (1-3), (2-4), (3-5), (4-2), (5-3). The values 2 and 4 are assigned to one object, while 1, 3 and 5, as 3 and 5 are matched pairs and 1 is associated with 3, so becomes part of that object. It is these class objects S_{Co}, that are used within the inter-sequence processing. This permits class objects from preceding sequences $\tilde{S_{Co}}$ to be compared with current ones, again looking for similarity with the binary Jaccard coefficient

$$J(S_{Co}, \tilde{S_{Co}}) = \arg\max_{S_{Co}} \frac{S_{Co} \wedge \tilde{S_{Co}}}{S_{Co} \vee \tilde{S_{Co}}} \tag{6}$$

These class objects within the inter-sequence then allow sequence to sequence continuity, thus allowing tracking of objects permitted that they maintain a level of feature similarity.

As the original SpikeSEG network outputs a semantic segmentation output, the HULK and SMASH processes are supplementary to this process in transforming the semantic regions into instance-based objects. Based on the block diagram seen in Fig. 1, the pseudo-code for the SMASH process is provided within Algorithm 1. This pseudo-code allows further insight to the method for comparison both intra- and inter-sequence and serves to compliment the block diagram seen in Fig. 1, but with the addition on the internal functions used to calculate the values.

From the pseudo-code, it can be determined that the SMASH method has two conditions before it concludes that instances are of the same object:

- That the two instances must have overlapping bounding boxes
- That the two instances must share a featural-temporal data

Without either of these, the multiplication of the similarity score and intersection over union will result in 0, and it will determine that the two instances are different objects. All instances that combine into the same object are then stored

Algorithm 1 SMASH

procedure INTRA-SEQUENCE(a, b)
  for *each input sequence* **do**
    for *each spiking instance in the classification layer* **do**
      perform HULK for each instance
      compute max and min, x and y values
      compute the $[X, Y, W, H]$ *bounding boxes*
      perform ASH to create S_{Ci}
      return S_{Ci}
    end for
    for *each hashed instance,* S_{Ci} **do**
      compute Similarity score against other instances, $S_{Ci'}$ *as in (2)*
      compute Bounding Box IoU score against other instances $S_{Ci'}$ *as in (3)*
      compute the SMASH score as in (4)
      assign max SMASH score instance pair to object S_{Co} *as in (5)*
      return $ListofS_{Co}$
    end for
  end for
end procedure
procedure INTER-SEQUENCE(a, b)
  for each class object S_{Co} **do**
    compute Similarity score against previous class objects $\tilde{S_{Co}}$ *as in (6)*
    return $max(J(S_{Co}, \tilde{S_{Co}}))$
  end for
end procedure

as one set of features and used at the inter-sequence stage to track objects over multiple input sequences.

The HULK and SMASH processes are able to build upon the semantic segmentation abilities of the semantic segmentation network. Giving object detection and instance segmentation capability, without the use of a regression sequence. Instead, this method opts for locality and similarity of the most salient features, utilising the information already available from the SCNN process.

3 Experimental Results

In the evaluation of the performance of the novel HULK SMASH algorithm presented in this paper, a number of tests have been carried out. The tests include:

- Semantic to instance segmentation with object detection
- Sequence to sequence tracking of objects with object occlusion and recovery
- Similarity matching for intra-class grouping (feature detection)

These tests are all carried out with the N-Caltech Dataset [35], mostly utilising the Face category. This is because Face subset provides a subtle yet distinct amount of intra-class variance, where each person clearly define this intra-class subset, the other classes don't have such easily definable subsets. This class within the dataset allows the successful extraction of multiple diverse face like features within the second convolution layer, which in turn provides a robust and general face detection within the classification layer. The testing was carried out using 27 of the 29 of the different people within the Face class, this is due to two of the people only appearing once in the set. With the 27 individuals in the dataset selected this gives 370 spiking sequences to work with. Each input sequence from the dataset containing 300ms of spiking activity, 100ms for each saccade in a triangle movement. 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, 10 ms of event data is buffered into 10 steps, representing 1 ms each (this value of 10 ms is chosen to empirical testing and based on the input spike count of the N-Caltech Dataset). This feature is used in testing to compare how HULK SMASH performs on short integration times, highlighting the sparse processing ability. The network parameters are set to be the same as within [33], with the only difference being 36 features available for the one class present. This was to limit the number of external factors and focus the testing on the intra-classification abilities rather than inter classification. To further validate the SMASH method the 5 and 10 class networks where also utilised from [34]. These tests helps to validate the SMASH method in a more complex feature similarity environment, where each class has less representation in layer Conv 2 neurons, as these networks utilised 16 features for of classes, so 80 and 160 neurons for the 5 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. The results contained show quantitative results for object detection, while providing qualitative results for the instance segmentation. This is partially due to the complex nature in which the event streams vary over time due to the change detection of the sensor, meaning a ground truth of what is the object and what is background or sensor noise is almost impossible or impractical to implement.

3.1 Semantic to Instance Segmentation with Object Detection

A series of multiple input streams are given as input to the SpikeSEG network, illustrated in Fig. 5. Where all the streams start at the same time, but a spiking events are displaced over a large spatial plane. The network then creates both the semantic segmentation output and via the HULK-SMASH process, an object detection/instance segmentation output. The input sequence is shown in Fig. 5 (a), with 5 input stream presented at the same times, 3 with Faces and 2 without. Fig. 5 (b) highlights the semantic segmentation output of the SpikeSEG network, while Fig. 5 (c) demonstrates the HULK process extracting each instance from the semantic representation. Fig. 5 (d) illustrates how the SMASH process groups instances that score above 0 into class instance objects. Lastly Fig. 5 (e) presents the final instance segmentation output, thanks to the object detection separating the semantic representation.

Within this scenario shown in Fig. 5, the separate input streams are not overlapping, consequently meaning the

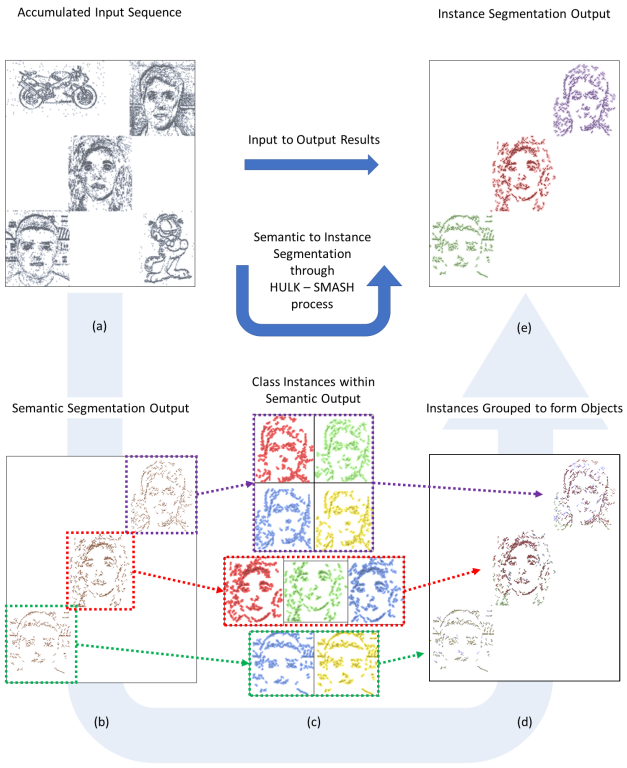

Fig. 5. Processing from Semantic Segmentation to Instance Segmentation: (a) the accumulated input sequence, (b) the semantic segmentation output, (c) the class instances from the semantic segmentation, (d) the grouped class objects and (e) the instance segmentation output

bounding boxes between input stream also do not overlap. This means the SMASH score between the input streams will always be 0 regardless of the similarity score. This is partially the intention of the SMASH process, allowing objects that do not share spatial locality to lead to false object grouping. As such the performance of the object detection to identify that the individual class instances belonging to one object is being evaluated, similar to the process depicted in Fig. 3 and 4. This means that even though the network is classifying all the Faces as just that, Faces, the SMASH process is able to identify that these do no share the same featural content. The object detection accuracy to identify three separate distinct faces is 100% over the complete input sequence, with numerical results tabulated within Table 1. However, over each input buffer, this drops to 96.14% as there are a few cases in which the similarity is 0 due to a low input spiking rate not producing the minimal number of spikes for activation over the short integration period. This is typical of the inputs from the start and end of the saccade movement within the N-Caltech dataset [35].

3.2 Object Detection and Segmentation in a Multi-Class Environment

To further evaluate the SMASH method, the multi-class semantic segmentation networks were tested. This allowed testing of the object detection and instance segmentation from the 5 and 10 class semantic segmentation environment.

TABLE 1
Results from the experimental testing split into 4 sections covering the Object detection, detection with occlusion, occlusion recovery and object self matching. *MS - Multistream

	Object Detection Accuracy (%)	
	per Input	per Sequence
MS N-CalTech (Face)	96.14	100
MS N-CalTech (5 Class)	95.17	100
MS N-CalTech (10 Class)	91.67	100

	Detection Accuracy with rate of Occlusion (%)		
	5%	25%	50%
MS N-CalTech (Face)	97.33	62	28

	Occlusion Recovery Rate (%)	
	without noise	with noise
MS N-CalTech (Face)	100	100

	Object Self Matching		
	Top 1	Top 3	Top 10
MS N-CalTech (Face)	95	100	100

Together with the increased number of classes present in each segment, there was also a reduction in the overall number of features available within the second convolution layer, Conv 2. This presents a scenario where the features are more generalised and therefore present less opportunities to learn features that help to diversify. Despite this, the SMASH process was still able to deliver high accuracy in determining how many object instances of a class there were, with 95.17% within the 5 class testing and 91.67% for the 10 class test, collated in Table 1. To help visualise the results from this experiment, two examples of successful instance segmentations outputs from the 10 class test are shown in Fig. 6, which shows for both (a) and (b), the input overlaid with the classification layer feature map and the semantic segmentation output respectively. Fig. 6 also indicates how the class instances are assigned in the 'Instance Map', where the numerical assignment is displayed to help illustrate the SMASH score also seen within the image. The Pairs are shown in numerical order with associated SMASH score, and how that is grouped into objects.

The classes other than Face presented a different problem to the Face class, as most of the other classes (excluding Stop Sign) had a higher feature count in the second convolution layer Conv 2. This presented a problem in which the timing of the occurrences of the features, was more important than its occurrence. Which during the Face only testing, was not an issue due to the sparsity at which those features appeared. This testing highlights the importance of the temporal component of the similarity matching, as simple feature-wise similarity matching alone is unable to determine class instances from one another. This extra temporal dimension can be seen a being able to leverage the saliency (timing) of features within individual class instances.

3.3 Object Occlusion

To help test the HULK-SMASH's ability to be able to perform sequence to sequence tracking of objects, further evaluation of the object detection capability is tested within three more challenging scenarios, where the class instances will now overlap. Taking the same multi-stream approach as the previous test, however, now the central stream of the image will be positioned such that occlusion of 5%,

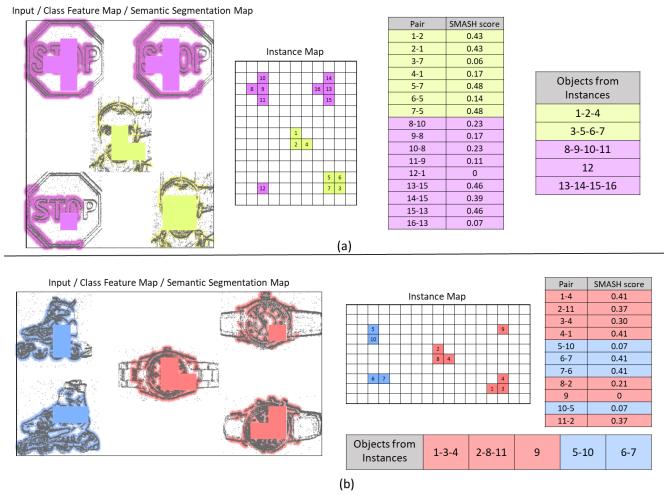

Fig. 6. Illustrating the process of going from semantic segmentation to instance wise object construction, displaying the instance mapping and pairing with SMASH scores that help to construct the class instance objects for two examples (a) and (b). Respectively showing the process for the classes Face and Stop Sign, and Watch and In-line Skate.

25% and 50% will occur. Partial occlusions test the feature extraction and similarity matching performance when there is a bounding box overlap, meaning the SMASH score is now the active measure of the combined similarity and locality. The three occlusion states are illustrated in Fig. 7, where (a), (b) and (c) depicted the occlusion covering for 5 %, 25% and 50% respectively as the stream of data in the centre is repositioned to create the occlusion. This was tested 5 times for each of the 27 faces with, pairing each face with 5 randomly selected face for each test (including the same face). Resulting in a total of 135 scenarios for each of the occlusion percentages.

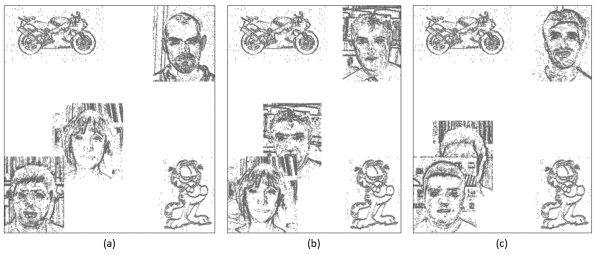

Fig. 7. Input occlusion example illustrating: (a) 5% occlusion of center image, (b) 25% occlusion of center image, and (c) 50% occlusion of center image

With the 5% occlusion, the performance was unchanged at 97.33% also shown in Table 1, which was to be expected considering the example shown in Fig. 7 (a) shows no overlapping of the facial features, with all images in the dataset having the faces centred in the image. While the 25% occlusion reduced the performance down to 62% and with 50% occlusion getting 28% also both in Table 1 when testing over each input buffer. A more detailed example from the 25% occlusion testing is shown in Fig. 8. The figure highlights the process from the input in (a), the class latent space neurons that were active in (b), the HULK end process of the active class pixels represented back in the pixel space with (c), and finally, the instance segmentation output (d), with three successful instances of each face detected. The only concern with this approach is the overlapped section between the bottom two images has a region that is assigned to two objects. This results in an area of uncertainty and even with a process where the two objects could compete for the pixels, the misclassification appears to exist in the feature detection domain, not the HULK-SMASH process. However, it must be noted that the failures within this testing phase are exclusively from the failure to find enough features to give a positive classification, not the incorrect assignment of instances. This element is highlighted in Fig. 9, where the classification process has failed to result in classification for the centre image; however, the bottom left and top right faces are correctly classified and segmented without overlap onto the unclassified region.

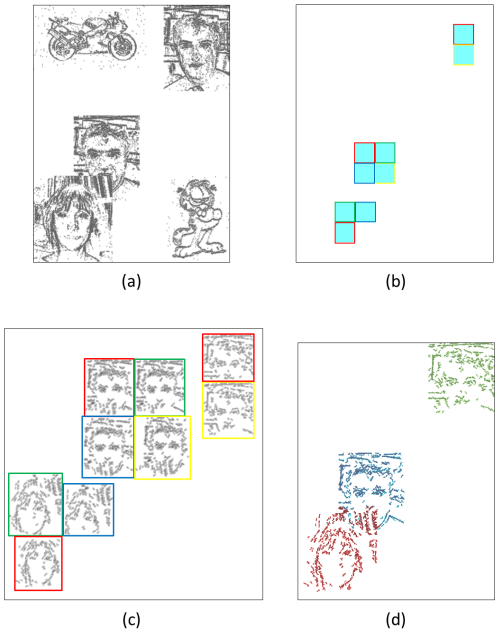

Fig. 8. Input 25% occlusion example illustrating: (a) The input sequence, (b) The latent classification space, (c) The class instance breakdown and (d) The instance segmentation output

3.4 Object Occlusion Recovery

As shown in the previous section, the object detection and instance segmentation method can identify features belonging to a particular instance of that class on an input by input basis, known as intra-sequence detection. The experimental set up for this section allows testing of object detection across inputs multiple sequences known as inter-sequence detection. This test then allows sequence to sequence tracking, then partial and full occlusion and subsequent recovery. The inputs are the full 300 ms sequence of the N-Caltech dataset [35] broken into the usual 10 ms steps as seen in previous sections. Each of the 27 faces was systematically tested against 3 versions of the other 26 faces (sequences for each face chosen at random), for a total of 2106 trials. This places two inputs at the top right and bottom left,

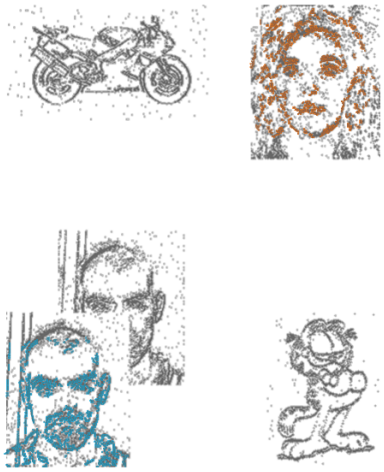

Fig. 9. Segmentation mapped over the input sequence, with a failure to identify the central image, thus failure to include in instance segmentation, however successful segmentation of the other two faces

then has then move diagonally across towards each other for each buffered input. The results of the test from a few selected instances are shown in Fig. 10. Fig. 10 (a) depicts the start position of the two different sequences and their respective instance segmentation and bounding boxes (red and black). Fig. 10 (b-e) show the transition of the respective segmentations diagonally across, with (c) and (d) showing the occlusion due to overlap where the black object completely occludes the red object. Fig. 10 (e) then highlights the detection picking up the red object again, from finding a high similarity with the red object that existed prior to the occlusion.

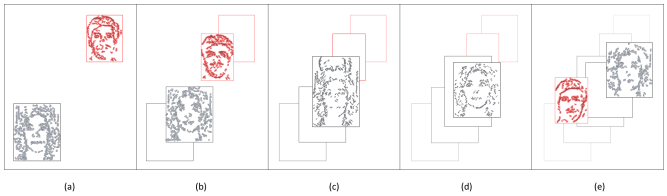

Fig. 10. Segmentation mapped over the input sequence, with a failure to identify the central image, thus failure to include in instance segmentation, however successful segmentation of the other two faces

The experimental results show the inter-sequence similarity matching is able to recover the occluded input sequence, such that across all the test sequences the similarity check managed to recover the occluded objects 100% of the time, as seen in Table 1. This feat is more impressive due to the occluded object being a dynamic sequence itself, meaning the post occlusion object is not just a replica of pre occlusion one. This occlusion recovery was also tested again with a noisy input, that is added noise to both the input sequence and to the general background. This noisy sequence is shown in Fig. 11, where Fig. 11 (a) and (e) shown the pre and post occlusion states, presenting that the recovery works correctly, while Fig. 11 (b), (c) and (d) show some of the issues the occlusion can cause. Fig. 11 (b) highlights the failure of the object detection in finding extra objects that do not exist. Fig. 11 (c) correctly illustrates the full occlusion state where only the female face object is present. Fig. 11 (d) displays the point at which partial occlusion is still causing only one object to be present, due to the similarity of class instances being close across the 9 instances in this case. Some of the 9 instances include multiple features of both objects, similar to what is shown in Fig. 11 (b). However, in this case, there was at least one instance that matched another within both objects. The object in Fig. 11 (d) is also blue as it scored the highest similarity with that object on the inter-sequence process. Despite some of the mentioned inaccuracies associated with the added noise, the system was able to re-identify occluded objects 100% of the time, as shown in Fig. 11 with this result recorded with the others in Table 1.

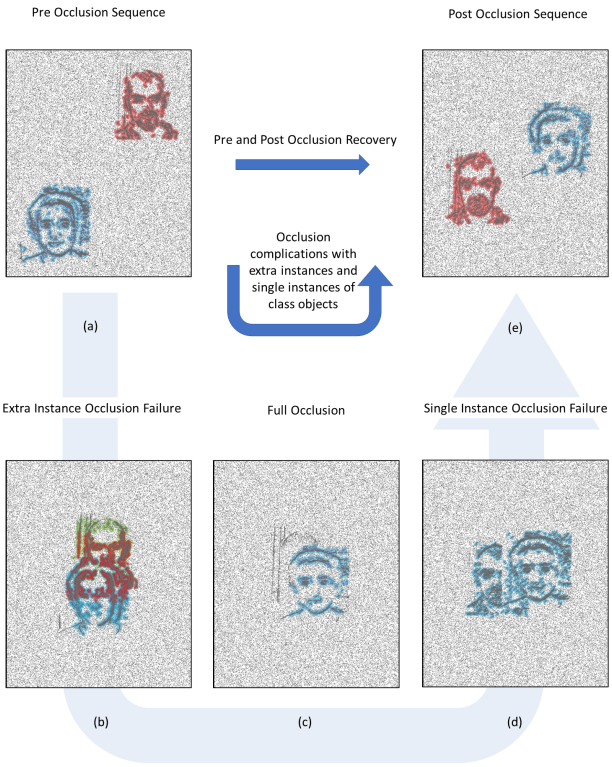

Fig. 11. Occlusion recovery with noisy input. (a) Pre Occlusion state, (b) Occlusion causing extra objects to appear, (c) Full occlusion with only one object present, (d) Partial occlusion grouping of objects and (e) Post occlusion state

4 Intuition and Understanding through Visualisation

The principle behind how object detection and instance segmentation are resolved can be better understood through visualisation. Through visualisations of the weights as they are mapped back into the pixel space through the subsequent layers are shown in Fig. 12. Fig. 12 (a) showing the features of Conv 1, which are the unlearned pre-determined

edge detection features, (b) showing the features of layer Conv 2 (Trans-Conv 2) and (c) highlighting the classification layer feature which depicts a Face. Fig. 12 (b) contains the bulk of the information used for the similarity matches process, thus contains the key to interpreting how the network can differentiate between different people in the dataset.

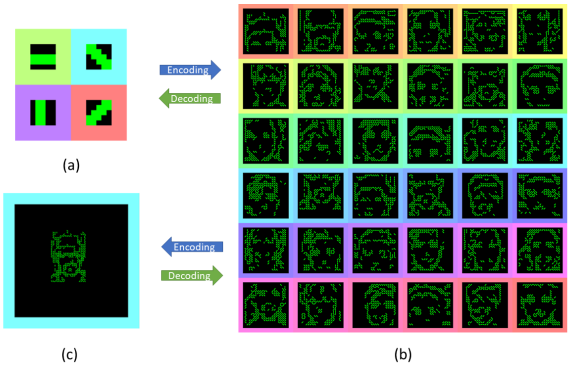

Fig. 12. The set of features from the trained network, (a) Conv 1, (b) Conv 2 and (c) Conv3

Mapping the features of the network from Conv 2 layer onto some original images from the Caltech dataset presents an insight into how this layer is differentiating between the different people in the dataset, as each person has unique and distinctive enough features to be learned through repeat occurrence. This ability to learn sub-classification feature clustering stems from the ability to classify up to 10 different objects due to the variance between the classes (inter-class) being higher than the variance of one particular class (intra-class). However, in this case, it is now the intra sub-class variance that allows the unique featural-temporal identifiers of each face to be captured. This is highlighted in Fig. 13 where 4 distinctive features that appear to represent a particular person from the dataset are overlaid onto the non-spiking version of the original image (for easier visualisation) it matches closest. In which the features of Fig. 13 appear to pick out the typical facial features of eyes nose mouth and hair.

To test just how well the features are allowing the individuals within the dataset to be recognised, an experiment matching each sequence with its top 1, top 3 and top 10 matches was carried out. This permits an insight into whether or not the HULK-SMASH process was able to differentiate between the people within the dataset, purely on re-occurrence of similar salient features, or in other words distinctive facial features in a featural-temporal manner. The test processes each buffered input, within each sequence of the testing set, and compares it with the full test set, recording the top, top 3 and top 10 most similar spatial-temporal patterns to its own. The results show that over 95% of the test inputs for an individual match the highest with themselves. While 100% match with themselves if extended into the top 3 and top 10, displayed in table 1. This result is illustrated in Fig. 14, where the top 10 similarity matches for a selection of inputs are shown in a coloured column-wise manner, with the left of the column showing the input sequence and the right showing the top 10 matches in descending order.

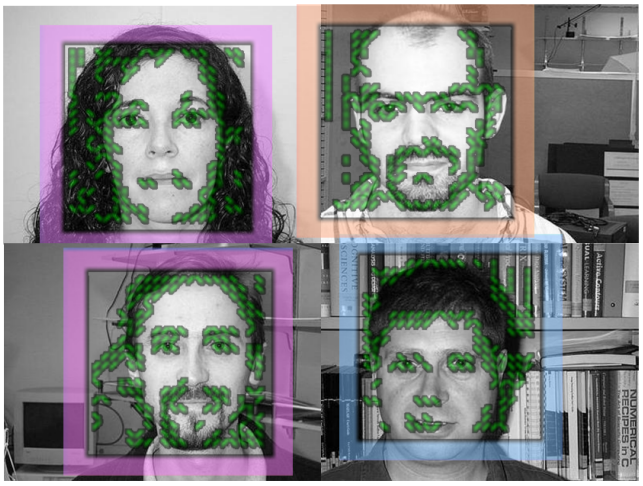

Fig. 13. Overlaid Features from Conv 2 onto Images from original Caltech Dataset, displaying how the learned features represent different faces

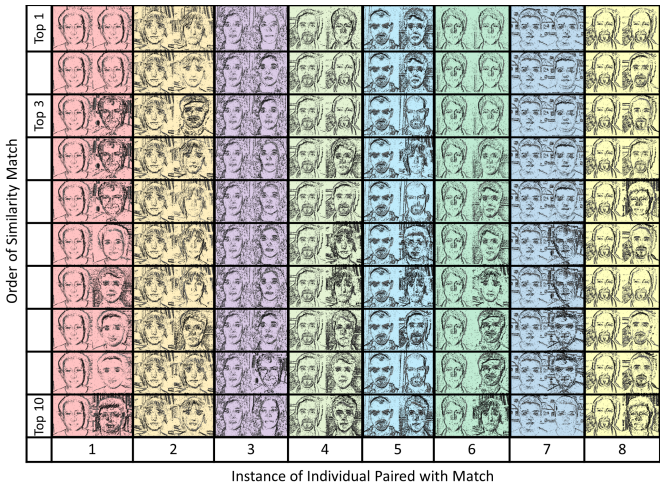

Fig. 14. Columns of top 10 matches, left on each column is the test input repeated 10 times with each of the 10 matches to the right, ranked in descending order. Each colour represents a new column for easier visual separation

Fig. 14 helps to build interpretability to the similarity matching process, allowing an understanding as to how and why the accuracy of the classification and object detection is as such. The sparse feature sets of a winner takes all approach to STDP allowing easier visualisation. This in turn helped to construct a method of matching these spatial-temporal features which are a result of the neuromorphic input to the spiking network both maintaining the important temporal causality of salient features.

From the experimental work carried out it is believed that the initial use of STDP for feature extraction plays a pivotal role in the in how the latent space spiking information of the SPikeSEG network is able to be utilised by HULK SMASH. The sparisty enforced by the winner take all STDP approach to learning features. Combined with the encoding of saliency into earlier spiking, thus giving it a higher likelihood to be learned by STDP. Together both of these features allow a unique learned representation of the

event data to be gained. Methods using supervision and backpropagation would be would have to try and enforce this sparsity and implement a loss function to capture this temporal saliency relationship. As typically they have a more dense and rate-based spiking internal representation. The sparse and temporal nature seem to be key components of the STDP learning that are often overlooked when comparing to supervised methods. Furthermore, the sparse edge-like feature representation and temporal connection to the input make methods like STDP more advantageous for event-driven neuromorphic sensing. Overall the feature representation appear to have less fidelity i n t he spatial domain, when compared to supervised methods. However, with HULK SMASH gains a high fidelity l atent temporal resolution, but looking at the SpikeSEG network's sparse internal representation. Further research could even incorporate HULK SMASH's instance segmentation output into a semi-supervised learning system to help add a third factor to the STDP learning, helping to offset this weak spatial feature extraction.

5 Conclusion

In this paper a new method for spiking object detection and instance segmentation, HULK-SMASH was presented. The system utilised STDP learned features that previously gave a semantic segmentation output. The classification layer was subsequently processed by unravelling each individual latent spiking neuron in the layer back to the pixel space, seen as HULK. This allows each instance of the classification layer to be treated as a class instance. It is then through comparing the class instances that objects can be identified through the SMASH score looking at the similarity and locality of instances. Permitting object detection and instance segmentation to be carried out on each buffered input. Then through similarity matching of the objects in sequences, sequence to sequence object detection and segmentation was achieved. This allowed object occlusion and reappearance to be realised, with the ability to deal with subtle changes to the object pre and post occlusion. The same mechanisms for SMASH could be used for continual online learning in which new objects are created so long as they trigger a classification n euron. V isualisation o f e xperimental results gave a better understanding and intuition as to where the SpikeSEG network was working well and why some failures occur. The HULK-SMASH processes were shown to be robust to multiple network instances with single and multi-class experiments with successful results while aiding in neural network interpretability. It is hoped that this method can help to extend tracking research based on using the event sensor and SNNs for processing.

References

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[2] S. J. Dickinson, A. Leonardis, B. Schiele, and M. J. Tarr, *Object categorization: computer and human vision perspectives*. Cambridge University Press, 2009.

[3] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2. Ieee, 1999, pp. 1150–1157.

[4] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[5] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1. Ieee, 2005, pp. 886–893.

[6] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[7] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Computer Vision, IEEE International Conference on*, vol. 3. IEEE Computer Society, 2003, pp. 1470–1470.

[8] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *European conference on computer vision*. Springer, 2010, pp. 143–156.

[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[10] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, 2018.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[14] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in neural information processing systems*, 2016, pp. 379–387.

[15] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3119–3127.

[16] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4293–4302.

[17] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2015, pp. 3431–3440. [Online]. Available: http://ieeexplore.ieee.org/document/7298965/

[18] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, dec 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7803544/

[19] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[20] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[21] G. Marcus, "Deep learning: A critical appraisal," *arXiv preprint arXiv:1801.00631*, 2018.

[22] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "The computational limits of deep learning," *arXiv preprint arXiv:2007.05558*, 2020.

[23] E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grahn, "Estimation of energy consumption in machine learning," *Journal of Parallel and Distributed Computing*, vol. 134, pp. 75–88, 2019.

[24] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3987–3995.

[25] K. Patel, E. Hunsberger, S. Batir, and C. Eliasmith, "A spiking neural network for image segmentation," *arXiv preprint arXiv:2106.08921*, 2021.

[26] Y. Kim, J. Chough, and P. Panda, "Beyond classification: Directly training spiking neural networks for semantic segmentation," *arXiv preprint arXiv:2110.07742*, 2021.

[27] D. Zheng, X. Lin, and X. Wang, "Image segmentation method based on spiking neural network with adaptive synaptic weights,"

in *2019 IEEE 4th International Conference on Signal and Image Processing (ICSIP)*. IEEE, 2019, pp. 1043–1049.
[28] C. M. Parameshwara, S. Li, C. Fermüller, N. J. Sanket, M. S. Evanusa, and Y. Aloimonos, "Spikems: Deep spiking neural network for motion segmentation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 3414–3420.
[29] G.-q. Bi and M.-m. Poo, "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type," *Journal of neuroscience*, vol. 18, no. 24, pp. 10 464–10 472, 1998.
[30] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "STDP-based spiking deep convolutional neural networks for object recognition," *Neural Networks*, vol. 99, pp. 56–67, mar 2018. [Online]. Available: www.sciencedirect.com/science/article/pii/S0893608017302903
[31] T. Masquelier and S. J. Thorpe, "Unsupervised learning of visual features through spike timing dependent plasticity," *PLoS computational biology*, vol. 3, no. 2, 2007.
[32] P. Kirkland, G. Di Caterina, J. Soraghan, Y. Andreopoulos, and G. Matich, "Uav detection: a stdp trained deep convolutional spiking neural network retina-neuromorphic approach," in *International Conference on Artificial Neural Networks*. Springer, 2019, pp. 724–736.
[33] P. Kirkland, G. Di Caterina, J. Soraghan, and G. Matich, "Spikeseg: Spiking segmentation via stdp saliency mapping," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.
[34] P. Kirkland, G. Di Caterina, J. Soraghan, and G. Matich, "Perception understanding action: adding understanding to the perception action cycle with spiking segmentation," *Frontiers in Neurorobotics*, vol. 14, 2020.
[35] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in neuroscience*, vol. 9, p. 437, 2015.
[36] P. Jaccard, "Distribution de la flore alpine dans le bassin des dranses et dans quelques régions voisines," *Bull Soc Vaudoise Sci Nat*, vol. 37, pp. 241–272, 1901.

Paul Kirkland completed both his PhD in Neuromorphic Engineering and MEng in Electrical and Mechanical Engineering at the University of Strathclyde. He is currently working as a researcher and deputy director of the Neuromorphic Sensor Signal Processing Lab, Electrical and Electronic Engineering, University of Strathclyde, Glasgow, United Kingdom. His interest areas are Neuromorphic algorithms, spiking neural networks and event-driven sensing.

Davide Manna (GS Member, IEEE) received both his M.Sc. degree in Software Engineering and his B.Sc. degree in Computer Engineering from Polytechnic of Turin , Italy, respectively in 2019 and 2017. From 2019 to 2020 he was with Tallinn University of Technology, Tallinn, Estonia, as a Deep Learning Researcher. He is currently working toward the PhD degree in the Neuromorphic Sensor Signal Processing Lab, University of Strathclyde, Glasgow, United Kingdom. His research interests include spiking neurons, unsupervised learning, neuromorphic algorithms and few-shots learning.

Alex Vicente (GS Member, IEEE) received the master's degree from Autonomous University of Barcelona, Barcelona ,Spain, in 2018 and the bachelor's degree from Pompeu Fabra University, Barcelona ,Spain, in 2017. From 2018 to 2020 he was with WheelRight Ltd, Oxford, United Kingdom. He is currently working toward the PhD degree in the Neuromorphic Sensor Signal Processing Lab, University of Strathclyde, Glasgow, United Kingdom. His research interests include deep learning, neuromorphic computing and spiking neural networks.

Gaetano di Caterina (Member, IEEE) Dr Gaetano Di Caterina received his BEng degree in Computer Engineering from the University of Naples in 2005, and his MEng degree in Computer and Electronic System and PhD degree from the University of Strathclyde in 2009 and 2013 respectively. He is Senior Lecturer and Chancellor's Fellow at the University of Strathclyde and is the Director of the Neuromorphic Sensor Signal Processing Lab (NSSP) in the Electronic and Electrical Engineering Department, working on various applications of neuromorphic engineering and deep learning, with strong links with academia, industry and governmental institutions.