






# Hybrid SNN-ANN: Energy-Efficient Classification and Object Detection for Event-Based Vision

Alexander Kugele<sup>1,2</sup>(✉) , Thomas Pfeil<sup>1</sup>, Michael Pfeiffer<sup>1</sup> ,  
and Elisabetta Chicca<sup>2,3</sup> 

<sup>1</sup> Bosch Center for Artificial Intelligence, 71272 Renningen, Germany  
{alexander.kugele,thomas.pfeil,michael.pfeiffer3}@de.bosch.com

<sup>2</sup> Bio-Inspired Circuits and Systems (BICS) Lab, Zernike Inst Adv Mat,  
University of Groningen, Nijenborgh 4, 9747 Groningen, AG, The Netherlands  
e.chicca@rug.nl

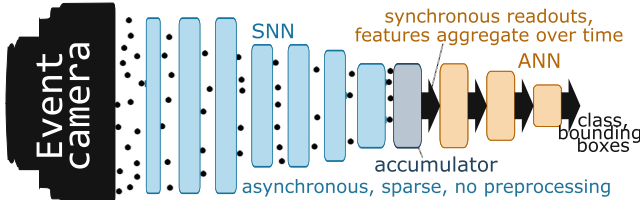
<sup>3</sup> Groningen Cognitive Systems and Materials Center (CogniGron),  
University of Groningen, Nijenborgh 4, 9747 Groningen, AG, The Netherlands

**Abstract.** Event-based vision sensors encode local pixel-wise brightness changes in streams of events rather than full image frames and yield sparse, energy-efficient encodings of scenes, in addition to low latency, high dynamic range, and lack of motion blur. Recent progress in object recognition from event-based sensors has come from conversions of successful deep neural network architectures, which are trained with backpropagation. However, using these approaches for event streams requires a transformation to a synchronous paradigm, which not only loses computational efficiency, but also misses opportunities to extract spatio-temporal features. In this article we propose a hybrid architecture for end-to-end training of deep neural networks for event-based pattern recognition and object detection, combining a spiking neural network (SNN) backbone for efficient event-based feature extraction, and a subsequent classical analog neural network (ANN) head to solve synchronous classification and detection tasks. This is achieved by combining standard backpropagation with surrogate gradient training to propagate gradients inside the SNN layers. Hybrid SNN-ANNs can be trained without additional conversion steps, and result in highly accurate networks that are substantially more computationally efficient than their ANN counterparts. We demonstrate results on event-based classification and object detection datasets, in which only the architecture of the ANN heads need to be adapted to the tasks, and no conversion of the event-based input is necessary. Since ANNs and SNNs require different hardware paradigms to maximize their efficiency, we envision that SNN backbone and ANN head can be executed on different processing units, and thus analyze the necessary bandwidth to communicate between the two parts. Hybrid networks are promising architectures to further advance machine learning approaches for event-based vision, without having to compromise on efficiency.

**Supplementary Information** The online version contains supplementary material available at [https://doi.org/10.1007/978-3-030-92659-5\\_19](https://doi.org/10.1007/978-3-030-92659-5_19).

**Keywords:** Object detection · Event-based vision · Spiking neural networks

## 1 Introduction



**Fig. 1.** Hybrid SNN-ANN models consist of an SNN backbone to compute features directly from event camera outputs which are accumulated and processed by an ANN head for classification and object detection. Using sparse, binary communication (dots) instead of dense tensors (arrows) to process events enables efficient inference. SNN and ANN can be on completely different devices.

Event-based vision sensors address the increasing need for fast and energy-efficient visual perception [10, 22, 24, 33, 41], and have enabled new use cases such as high-speed navigation [7], gesture recognition [26], visual odometry and SLAM [28, 46, 48]. These sensors excel at very low latency and high dynamic range, while their event-based encoding creates a sparse spatio-temporal representation of dynamic scenes. Every event indicates the position, precise time, and polarity of a local brightness change.

In conventional frame-based computer vision deep learning-based methods have led to vastly improved performance in object classification and detection, so it is natural to expect a boost in performance also from applying deep neural networks to event-based vision. However, such an approach has to overcome the incompatibility of machine learning algorithms developed for a synchronous processing paradigm, and the sparse, asynchronous nature of event-based inputs. Recent successful approaches for processing event data have therefore relied on early conversions of events into filtered representations that are more suitable to apply standard machine learning methods [1, 32, 44]. Biologically inspired spiking neural networks (SNNs) in principle do not require any conversion of event data and can process data from event-based sensors with minimal preprocessing.

However, high performing SNNs rely on conversion from standard deep networks [38, 40], thereby losing the opportunity to work directly with precisely timed events. Other approaches like evolutionary methods [30] or local learning rules like STDP [2, 16] are not yet competitive in performance.

Here we describe a hybrid approach that allows end-to-end training of neural networks for event-based object recognition and detection. It combines sparse spike-based processing of events in early layers with off-the-shelf ANN layers to

process the sparse, abstract features (see Fig. 1 for an illustration). This is made possible by combining standard backpropagation with recently developed surrogate gradient methods to train deep SNNs [4, 21, 29, 35, 42, 47]. The advantage of the hybrid approach is that early layers can operate in the extremely efficient event-based computing paradigm, which can run on special purpose hardware implementations [3, 5, 9, 27, 34, 39]. The hybrid approach is also optimized for running SNN parts and conventional neural networks on separate pieces of hardware by minimizing the necessary bandwidth for communication. In our experiments we demonstrate that the hybrid SNN-ANN approach yields very competitive accuracy at significantly reduced computational costs, and is thus ideally suited for embedded perception applications that can exploit the advantages of the event-based sensor frontend.

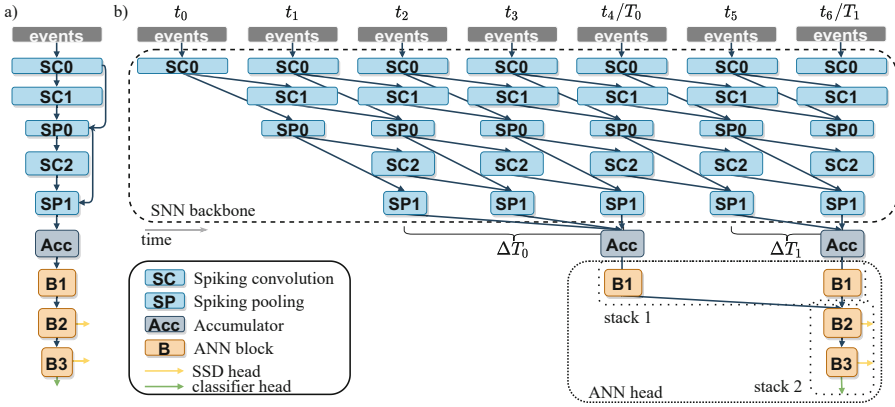
Our main contributions are as follows:

- We propose a novel hybrid architecture that efficiently integrates information over time without needing to transform input events into other representations.
- We propose the first truly end-to-end training scheme for hybrid SNN-ANN architectures on event camera datasets for classification and object detection.
- We investigate how to reduce communication bandwidths for efficient hardware implementations that run SNNs and ANNs on separate chips.
- We analyze how transfer learning of SNN layers increases the accuracy of our proposed architecture.

## 2 Related Work

A variety of **low level representations** for event-based vision data have been explored: The HOTS method [19] defines a time surface, *i.e.*, a two-dimensional representation of an event stream by convolving a kernel over the event stream. This method was improved in [44] by choosing an exponentially decaying kernel and adding local memory for increased efficiency. In [12], a more general take on event stream processing is proposed, utilizing general kernel functions that can be learned and project the event stream to different representations. Notably, using a kernel on event camera data and aggregating the result is the same as using a spiking neural network layer. Our approach allows learning a more general low level representation by using a deep SNN with an exponentially decaying kernel, compared to only one layer in [12] and learnable weights compared to [19, 44].

**Conversion approaches** such as [38, 40] transform trained ANNs into SNNs for inference, and so far have set accuracy benchmarks for deep SNNs. Conversion methods train on image frames and do not utilize the membrane potential or delays to integrate information over time. In [18] networks are unrolled over multiple time steps, which allows training on event camera datasets. However, temporal integration is only encoded in the structure of the underlying ANN, but not in the dynamics of spiking neurons. In their formulation, the efficiency of the SNN is limited by the rate coding of the neurons, which is potentially



**Fig. 2.** Training a hybrid network with a DenseNet backbone. a) Compact representation. b) Network rolled out over time steps. The SNN backbone computes sparse features from event camera data that are accumulated at time intervals  $\Delta T_i$ . The ANN head integrates features from multiple outputs (2, in this example) for a prediction (classification or object detection). We use a time step of 1 ms to integrate information over small time scales. During inference, the SNN backbone runs asynchronously without a time-stepped rollout, enabling potential savings in computation. Layers with the same name share weights.

more inefficient than encodings learned via end-to-end training in our hybrid SNN-ANN approach. In addition, conversion methods typically do not optimize for energy-efficiency.

**SNN training with variants of backpropagation** has been demonstrated in [21, 42, 47], albeit on simpler architectures (*e.g.*, only one fully-connected layer in [21]) and, in general, without delays during simulation. A mixed approach is used in [35], first training and converting an ANN and then training the converted SNN. Our approach uses synaptic delays and skip connections, exploring how complex ANN architectures translate to SNN architectures. The closest architecture to ours is from [20], which is trained from scratch to predict optical flow. Their U-Net with an SNN encoder transmits information from all SNN layers, leading to a high bandwidth from SNN to ANN. We improve on this by using only a single layer to transmit information from SNN to ANN, and extend to applications in classification and object detection tasks.

**Conventional ANN architectures** are used in [36] to solve the task of image reconstruction from events with a recurrent U-Net, and they subsequently show that classification and object detection are possible on the reconstructed images. No SNN layers are used in this case, resulting in a computationally expensive network and the need to preprocess the event camera data. Faster and more efficient training and inference for object detection is presented in [32], who propose a recurrent convolutional architecture that does not need to explicitly

reconstruct frames. As the sparsity of the event stream is not utilized, it is expected that gains in energy-efficiency are possible with SNN approaches.

### 3 Methods

This section introduces the proposed hybrid SNN-ANN architecture and describes training, inference and metrics we used to evaluate our method. Our hybrid network consists of two parts: An SNN backbone that computes features from raw event camera inputs, and an ANN head that infers labels or bounding boxes at predefined times (see Fig. 2). The overall task is to find an efficient mapping from a sequence of event camera data  $E$  in a time interval  $T$  to a prediction  $P$ , which can be a label  $l$  or a set of bounding boxes  $B$ . Our approach consists of three stages: First, continuously in time, an intermediate representation  $I = S(E)$  is generated using the SNN backbone. Second, this intermediate representation is accumulated at predefined points in time. Third, when all accumulators are filled, the accumulated intermediate representations are mapped via the ANN head  $A$  to the final prediction  $P = A(\text{Acc}(I)) = A(\text{Acc}(S(E)))$ . The following sections describe all three parts in more detail.

#### 3.1 SNN Backbone

Spiking neural networks (SNNs) are biologically inspired neural networks, where each neuron has an internal state. Neurons communicate via spikes, i.e. binary events in time, to signal the crossing of their firing thresholds. Upon receiving a spike  $i$ , the synaptic current  $I$  changes proportionally to the synaptic weight  $w$ , which in turn leads to a change of the neuron’s internal state  $V$ . Because of the binary and sparse communication, these networks can be significantly more energy-efficient than dense matrix multiplications in conventional ANNs (see also [37]).

The task of the SNN backbone  $S$  is to map a sequence of raw event camera inputs  $E$  into a compressed, sparse, and abstract intermediate representation  $I = S(E)$  in an energy-efficient way. More concretely, the input is a stream of events  $e_i = (t_i, x_i, y_i, p_i)$ , representing the time  $t_i$  and polarity  $p_i$  of an input event at location  $(x_i, y_i)$ . Polarity is a binary  $\{-1, 1\}$  signal that encodes if the brightness change is positive (brighter) or negative (darker). This stream is processed by the SNN without further preprocessing. In our implementation, the first layer has two input channels representing the two polarities. In contrast to previous work such as [20] and [18], the input events are never transformed into voxel grids or rate averages, but directly processed by the SNN to compute the intermediate representation  $S(E)$ . The spiking neuron model we use is the leaky integrate-and-fire model [13], simulated using the forward Euler method,

$$I_i = \alpha I_{i-1} + \sum_j w_j S_j \quad (1)$$

$$V_i = \beta V_{i-1} + I_i \quad (2)$$

$$S_i = \Theta(V_i - V_{\text{th}}) \quad (3)$$

where  $V$  is the membrane potential,  $I$  is the presynaptic potential,  $S_{j,i}$  are the binary input and output spikes,  $V_{\text{th}}$  is the threshold voltage, and  $t_d = t_i - t_{i-1}$  is an update step. The membrane leakage  $\beta$  and the synaptic leakage  $\alpha$  are given as  $\exp(-t_d/\tau_{\text{mem}})$  and  $\exp(-t_d/\tau_{\text{syn}})$ , respectively. To simulate the membrane reset, we add a term to Eq. (2) which implements the reset-by-subtraction [38] mechanism,

$$V_i = V_i - V_{\text{th}}S_{i-1}. \quad (4)$$

The threshold is always initialized as  $V_{\text{th}} = 1$  and trained jointly with the weights (see appendix for details).

We simulate our network by unrolling it in time with a fixed time step  $t_d$ . Figure 2b shows the training graph for a rollout of 7 time steps. Our simulation allows choosing arbitrary delays for the connections of different layers, which determines how information is processed in time. Inspired by recent advances, we choose to implement streaming rollouts [8] in all our simulations. This means that each connection has a delay of one time step, in accordance with the minimum delay of large-scale simulators for neuroscience [45]. This allows integrating temporal information via delayed connections, in addition to the internal state.

The SNN backbone  $S(E)$  outputs a set of sequences of spikes in predefined time intervals  $T_{\text{out},i}$ :  $I = (e_j)_{t_j \in T_{\text{out},i}}$ . An example is shown in the dashed box of Fig. 2. Details about the backbones used can be found in Sect. 3.8. The figure shows an unrolled DenseNet backbone with two blocks (SC0 to SP0 and SP0 to SP1, respectively), where two output intervals are defined as  $\Delta T_0 = [t_2, T_0]$  and  $\Delta T_1 = [t_5, T_1]$ . This structure is used during training, where a time-stepped simulator approximates the continuous-time SNN. During inference, the SNN backbone runs asynchronously, enabling savings in computation.

### 3.2 Accumulator

Our model connects the sparse, continuous representation of the SNN with the dense, time-stepped input of the ANN with an accumulator layer for each output interval  $\Delta T_i$  (Acc in Fig. 2). The task of this layer is to transform the sparse data to a dense tensor. We choose the simple approach of summing all spikes in each time interval  $\Delta T_i$  to get a dense tensor with the feature map shape  $(c_f, h_f, w_f)$ .

### 3.3 ANN Head

The ANN head processes the accumulated representations from the accumulators to predict classes or bounding boxes. The general structure of the ANN head can be described with three parameters: the number of SNN outputs  $n_{\text{out}}$ , the number of stacks  $n_s$  and the number of blocks per stack  $n_l$ . The exemplary graph in Fig. 2 has  $n_{\text{out}} = 2$ ,  $n_s = 2$  and  $n_l = 2$ . Having multiple outputs and stacks allows to increase the temporal receptive field, *i.e.*, the time interval the ANN uses for its predictions (for details see appendix). All blocks with the same name share their weights to reduce the number of parameters. The number of blocks can be different for each stack. In most experiments we use

two stacks with 1 and 3 blocks, respectively. The dense representation for each  $\Delta T_i$  is then further processed by each stack, where results are summed at the end of a stack and used as input for the next stack. Each block in a stack consists of batch normalization BN, convolution C, dropout Drop, ReLU and pooling P, *e.g.*,  $B0(x) = P(\text{ReLU}(\text{Drop}(\text{C}(\text{BN}(x))))))$ . We use a convolutional layer with kernel size 2, stride 2, learnable weights and ReLU activation as pooling layer. Whenever we use dropout in conjunction with weight sharing, we also share the dropout mask. In the case of classification, a linear layer is attached to the last block in the last stack (see Sect. 3.6). For object detection, an SSD head is attached to selected blocks in the last stack (see Sect. 3.7).

### 3.4 Energy-efficiency

We use the same metric as [38], *i.e.*, we count the number of operations of our network. Due to the sparse processing and binary activations in the SNN, the number of operations is given by the synaptic operations, *i.e.*, the sum of all post-synaptic current updates. For ANN layers, we count the number of multiply-add operations, as the information is processed with dense arrays. For this metric, it is assumed that both SNN and ANN are run on dedicated hardware. Then the total energy is proportional to the number of operations plus a constant offset. We discuss benefits, drawbacks and alternatives to this metric in the appendix. To regularize the number of spikes we utilize an  $L_1$  loss on all activations of the SNN backbone

$$L_s = \sum_{l,b,i,x,y} \frac{\lambda_s}{LBTW_l H_l} |S_{l,i,b,x,y}|. \quad (5)$$

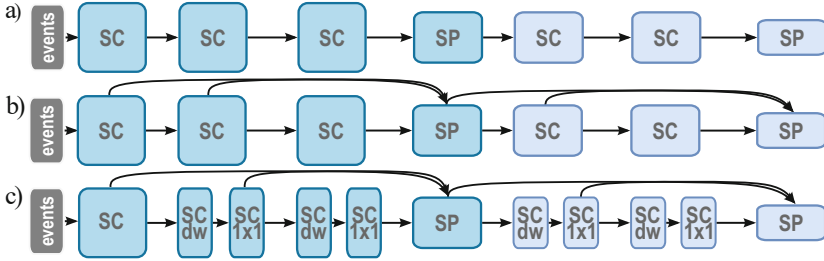
with a scaling factor  $\lambda_s$ , the number of layers  $L$ , the batch size  $B$ , total simulation time  $T$ , and width  $W_l$  and height  $H_l$  of the respective layer. This also reduces the bandwidth, as discussed in Sect. 3.5.

### 3.5 Bandwidth

As we expect that special hardware could be used to execute at least the SNN backbone during inference, we want to minimize the bandwidth between SNN and ANN to avoid latency issues. We design our architectures such that only the last layer is connected to the ANN and use  $L_1$  loss on the activations to regularize the number of spikes. This is in contrast to [20], where each layer has to be propagated to the ANN. We report all bandwidths in MegaEvents per second (MEv/s) and MegaBytes per second (MB/s). One event equals 6.125 Bytes, because we assume it consists of 32 bit time +16 bit spatial coordinates +1 bit polarity.

### 3.6 Classification

To classify, we attach a single linear layer to the last block in the last stack of our hybrid network. We use the negative log-likelihood loss of the output of



**Fig. 3.** The different backbones in compact representation with spiking convolutional (SC) and pooling (SP) layers. a) VGG. b) DenseNet. c) DenseSep (DenseNet with depthwise separable convolutions). Depthwise separable convolutions consist of a depthwise convolution (dw) and a convolution with kernel size  $1 \times 1$ . Shades of blue mark different blocks. All depicted networks have 2 blocks and 2 layers per block. Multiple inputs are concatenated.

this layer. Additionally, we use  $L_2$ -regularization (weight decay) with a factor of 0.001. We use the Adam optimizer [17] with a learning rate of  $\eta = 0.01$  and a learning rate schedule to divide it by 5 at 20%, 80% and 90% of the total number of epochs  $n_e = 100$ .

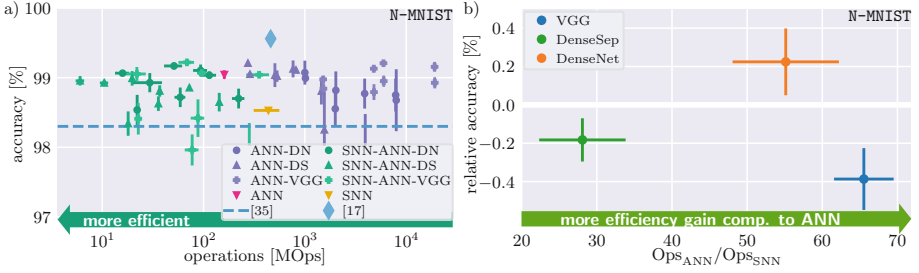
### 3.7 Object Detection

We use the SSD (single shot detection) architecture [25], which consists of a backbone feature extractor and multiple predictor heads. Features are extracted from the backbone at different scales and for each set of features, bounding boxes and associated classes are predicted, relative to predefined prior boxes. In the appendix, we present a novel, general way to tune the default prior boxes in a fast and efficient way. During inference, non-maximum suppression is used on the output of all blocks to select non-overlapping bounding boxes with a high confidence. The performance of the network is measured with the VOC mean average precision (mAP) [6]. If not otherwise denoted, we use the same learning hyperparameters as in Sect. 3.6 but a smaller learning rate of 0.001.

### 3.8 Backbone Architectures

Three different architecture types are used in our experiments: VGG [43], DenseNet [15] and DenseSep. A VGG network with  $N_b$  blocks and  $N_l$  layers per block is a feed-forward neural network with  $N_b \cdot N_l$  layers, where each output channel is given by  $g \cdot l$  with  $l$  the layer index (starting from 1). The DenseNet structure consists of  $N_b$  blocks, where each block consists of  $N_l$  connected layers per block. DenseSep is a combination of the depthwise separable block of MobileNet [14] and the DenseNet structure (see Fig. 3).





**Fig. 4.** a) Accuracy on the N-MNIST test set vs. number of operations for different architectures. Hybrid SNN-ANN architectures (green) overcome the efficiency limit of conversion-based architectures (blue diamond) with only a minor drop in accuracy. SNN-ANN architectures are more efficient than almost all ANNs (purple). Compared to the SNN and ANN baseline, our hybrid networks increase accuracy and energy-efficiency. b) Relative accuracy on the N-MNIST test set vs. relative number of operations. The VGG backbone has the highest gain in energy-efficiency, while losing significantly in accuracy. Other backbones show a minor decrease with a significant gain in energy-efficiency. We report the mean and error of the mean over 6 repetitions. (Color figure online)

## 4 Results

### 4.1 Classification on N-MNIST

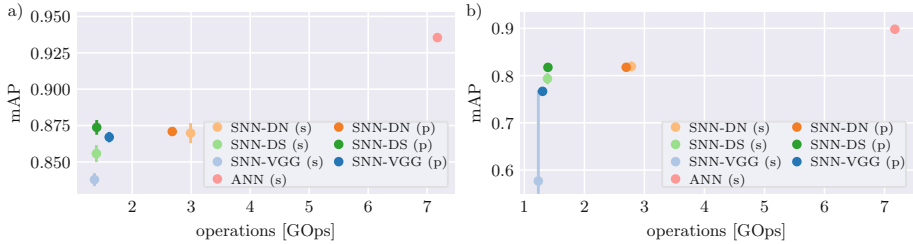
We train a hybrid SNN-ANN network for classification on the N-MNIST dataset [31]. In N-MNIST, each MNIST digit is displayed on an LCD monitor while an event camera performs three saccades within roughly 300 ms. It contains the same 60 000 train and 10 000 test samples as MNIST at a resolution of  $34 \times 34$  pixels. Here we compare the performance of our hybrid network to networks where the SNN backbone is replaced with ANN layers (ANN-ANN), two baselines and two approaches from the literature: A conversion approach [18], where a trained ANN is converted to a rate-coded SNN and an ANN that reconstructs frames from events and classifies the frames [36]. The first baseline is a feed-forward ANN with the same structure as our VGG SNN-ANNs, but where all time steps are concatenated and presented to the network as different channels. The second baseline is an SNN of the same structure, where we accumulate the spikes of the last linear layer and treat this sum as logits for classification. The SNN is unrolled over the same number of time steps than the SNN-ANNs. We report results for multiple network sizes and configurations to show that hybrid SNN-ANNs generally perform better in terms of energy-efficiency and bandwidth (detailed architecture parameters can be found in the appendix). Mean values of accuracy and number of operations are reported together with the error of the mean over 6 repetitions, using different initial seeds for the weights. For one DenseSep network only 5 iterations are reported, because training of one trial did not converge, resulting in a significant outlier compared to the performance of all other networks. In Fig. 4a, we show the accuracy on the N-MNIST test set vs. the

number of operations for our architectures, baselines and related approaches from the literature. Our hybrid networks (green, SNN) reach similar accuracies to the ANNs (purple), while being consistently more energy-efficient. The best hybrid architecture is a DenseNet with  $g = 16$ ,  $N_1 = 3$  and  $n_{\text{out}} = 1$ . Compared to [18], it performs slightly worse ( $(99.10 \pm 0.09)\%$  vs.  $(99.56 \pm 0.01)\%$ ) in accuracy, but improves significantly on the number of operations ( $(94 \pm 17)$  MOps vs.  $(460 \pm 38)$  MOps) despite having more parameters (504 025 vs. 319 890). The average bandwidth between SNN and ANN is  $(0.250 \pm 0.044)$  MEv/s for this architecture, or approximately 1.53 MB/s. Our smallest DenseNet with 64 875 parameters ( $g = 8$ ,  $N_1 = 2$ ,  $n_{\text{out}} = 1$ ) is even more efficient, needing only  $(15.9 \pm 2.5)$  MOps at a bandwidth of  $(0.144 \pm 0.034)$  MEv/s to reach  $(99.06 \pm 0.03)\%$  accuracy. Our results are mostly above [36] in terms of accuracy, although our networks are much smaller (their networks have over 10 M parameters). Due to the large parameter size, we also estimate that we should be more energy-efficient, although the authors do not provide any numbers in their publication.

In Fig. 4b, we plot the average per backbone over all architectures in Fig. 4a, relative to the averages of the ANN-ANN architectures. All hybrid SNN-ANNs improve the number of operations significantly over ANN-ANN implementations with at most a minor loss in accuracy. Hybrid DenseNets provide the best accuracy-efficiency trade-off with an average improvement of about a factor of 56 in energy-efficiency while also increasing accuracy by approximately 0.2%. For VGG architectures, the energy-efficiency is increased by a factor of roughly 65 at a loss in accuracy of 0.4 between hybrid and ANN-ANN architectures. Hybrid DenseSep architectures lose approximately 0.2 accuracy points, but gain a factor of about 28 in energy-efficiency. We assume that the DenseSep architectures perform the worst in comparison for two reasons: First, they are already the most efficient ANN architectures, so improving is harder than for less optimized architectures. Second, as the effective number of layers is higher in this architecture compared to the other two, optimizing becomes more difficult and gradient deficiencies (vanishing, exploding, errors of surrogate gradient) accumulate more. Hyperparameter optimization (learning rate, dropout rates, regularization factors for weights and activations) was not performed for each network separately, but only once for a DenseNet with  $N_1 = 2$ ,  $g = 16$ ,  $n_{\text{out}} = 1$  on a validation set that consisted of 10% of the training data, *i.e.*, 6000 samples.

## 4.2 Classification on N-CARS

N-CARS is a binary car vs. background classification task, where each sample has a length of 100 ms. We train the same networks as in Sect. 4.1 with a growth factor  $g = 16$  and compare to the same baselines and two results from the literature. We see the same trend in energy-efficiency improvements over ANNs, with factors ranging from 15 (DenseSep) to 110 (VGG). Relative accuracy decreases significantly for DenseSep by 1.5 points, but is 0.75 points higher for VGG. These results, together with the results in Fig. 4b suggest, that the more efficient an architecture already is, the less can be gained from training an equiva-



**Fig. 5.** Mean average precision (mAP) over number of operations for `shapes_translation-30/90` (a/b, mean over 4 trials). The architectures with the best results on the `N-MNIST` training are either trained from scratch (s) or initialized with the weights from the training (pretrained, p). Pretrained architectures improve over random initialization (VGG, DenseSep) or are on par with it (DenseNet).

lent hybrid SNN, although the gains of at least a factor of 15 in energy-efficiency is still significant. In comparison to our ANN and SNN baselines, our SNN-ANNs perform better in terms of accuracy and number of operations. Two out of four SNN runs could not go beyond chance level and were therefore excluded from the validation. In conclusion, using hybrid SNN-ANNs saves energy compared to SNNs and ANNs of similar structure. In this experiment, our architecture is not competitive to state-of-the-art [18] in accuracy, but has a similar energy demand with approximately double the number of parameters. Detailed results and figures can be found in the appendix.

### 4.3 Object Detection on `shapes_translation`

The `shapes_translation` dataset [28] contains frames, events, IMU measurements and calibration information of a scene of different shapes pinned to a wall. As training data for event-based vision object detection is scarce [11], we labelled bounding boxes for each of the 1356 images and 10 shapes, resulting in 9707 ground truth bounding boxes. A detailed description and an example of the dataset can be found in the appendix. We provide two train/test splits: `shapes_translation-90` where 90% of the data is randomly assigned to the train set and a more difficult split `shapes_translation-30`, where only 30% of the data is used in the train set. In section Sect. 4.4 and Sect. 4.5, we present the results on `shapes_translation-30/90`.

### 4.4 Results on `shapes_translation-90`

For all backbones, we take the architecture parameters from the best network in the `N-MNIST` task. See the appendix for details. We want to compare hybrid networks trained from scratch with networks where the SNN backbone is initialized with network weights trained on `N-MNIST`. This allows investigating the effect of transfer learning during training. Results are shown in Fig. 5. The networks with pretrained weights always converge to higher mAP than their non-pretrained

counterpart. The DenseNet and DenseSep backbones perform better than VGG. This is in agreement with the results for classical ANNs, where DenseNet architectures outperform VGG on image-based datasets like ImageNet. Our best network is a pretrained SNN-ANN DenseSep with a mean average precision of  $(87.37 \pm 0.51) \%$ ,  $(1398.9 \pm 2.3)$  MOps operations and a bandwidth of 11.0 MB/s. A comparable ANN backbone would require a bandwidth of 1210 MB/s. A regular SSD architecture with the same backbone as our VGG network, where all time steps are concatenated over the channels outperforms our networks in terms of mAP, but also needs significantly more operations. We report the detailed results in the appendix.

#### 4.5 Results on `shapes_translation-30`

We do the same evaluation as in Sect. 4.4, but with a 30/70% training/test data split (Fig. 5). The mAP is higher for networks with pretrained weights for DenseSep and VGG and on par with DenseNet. As with `shapes_translation-90`, the backbones with skip connections perform better than the VGG backbone. One of the four VGG experiments did not fully converge, explaining the high standard deviation. Our best network is an SNN-ANN DenseNet trained from scratch with  $(2790 \pm 50)$  MOps operations, a mean average precision of  $(82.0 \pm 1.0) \%$ , and a bandwidth of 2.68 MB/s. A comparable ANN backbone would have a bandwidth of 864 MB/s. As in Sect. 4.4 the regular SSD architecture is better in mAP but worse in the number of operations. We report the detailed results in the appendix.

## 5 Conclusion

In this paper, we introduced a novel hybrid SNN-ANN architecture for efficient classification and object detection on event data that can be trained end-to-end with backpropagation. Hybrid networks can overcome the energy-efficiency limit of rate-coded converted SNN architectures, improving by up to a factor of 10. In comparison to similar ANN architectures, they improve by a factor of 15 to 110 on energy-efficiency with only a minor loss in accuracy. Their flexible design allows efficient custom hardware implementations for both the SNN and ANN part, while minimizing the required communication bandwidth. Our SNN-ANN networks learn general features of event camera data, that can be utilized to boost the object detection performance on a transfer learning task.

We expect that the generality of the features can be improved when learning on larger and more diverse datasets. Our work is particularly suited for datasets where temporal integration happens on a short time interval, but struggles for longer time intervals, *e.g.*, multiple seconds due to the immense number of roll-out steps needed. Recent advances in deep learning, particularly C-RBP [23] can potentially help to overcome this by using recurrent backpropagation that has a constant memory complexity with number of steps (compared to backpropagation, where the memory-complexity is linear). This also would ensure that

our networks converge to a fixed point over time, potentially making predictions more stable. More work on surrogate gradients and methods to stabilize training can further help to increase both energy-efficiency and performance of hybrid networks. Our work is a first step towards ever more powerful event-based perception architectures that are going to challenge the performance of image-based deep learning methods.

**Acknowledgments.** The authors would like to acknowledge the financial support of the CogniGron research center and the Ubbo Emmius Funds (Univ. of Groningen). Furthermore, this publication has received funding from the European Union’s Horizon 2020 research innovation programme under grant agreement 732642 (ULPEC project).

## References

1. Amir, A., et al.: A low power, fully event-based gesture recognition system. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7243–7252 (2017)
2. Barbier, T., Teulière, C., Triesch, J.: Unsupervised learning of spatio-temporal receptive fields from an event-based vision sensor. In: Farkaš, I., Masulli, P., Wermter, S. (eds.) ICANN 2020. LNCS, vol. 12397, pp. 622–633. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-61616-8\\_50](https://doi.org/10.1007/978-3-030-61616-8_50)
3. Billaudelle, S., et al.: Versatile emulation of spiking neural networks on an accelerated neuromorphic substrate. In: 2020 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5 (2020). <https://doi.org/10.1109/ISCAS45731.2020.9180741>
4. Cramer, B., et al.: Surrogate gradients for analog neuromorphic computing. arXiv 2006.07239 (2021)
5. Davies, M., et al.: Loihi: a neuromorphic manycore processor with on-chip learning. *IEEE Micro* **38**(1), 82–99 (2018). <https://doi.org/10.1109/MM.2018.112130359>
6. Everingham, M., Gool, L.V., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (VOC) challenge. *Int. J. Comput. Vis.* **88**, 303–308 (2009). <https://www.microsoft.com/en-us/research/publication/the-pascal-visual-object-classes-voc-challenge/>, printed version publication date: June 2010
7. Falanga, D., Kleber, K., Scaramuzza, D.: Dynamic obstacle avoidance for quadrotors with event cameras. *Sci. Robot.* **5**(40) (2020). <https://doi.org/10.1126/scirobotics.aaz9712>
8. Fischer, V., Koehler, J., Pfeil, T.: The streaming rollout of deep networks - towards fully model-parallel execution. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) *Advances in Neural Information Processing Systems*, vol. 31, pp. 4039–4050. Curran Associates, Inc. (2018). <http://papers.nips.cc/paper/7659-the-streaming-rollout-of-deep-networks-towards-fully-model-parallel-execution.pdf>
9. Furber, S.B., et al.: Overview of the SpiNNaker system architecture. *IEEE Trans. Comput.* **62**(12), 2454–2467 (2013). <https://doi.org/10.1109/TC.2012.142>
10. Gallego, G., et al.: Event-based vision: a survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **1** (2020). <https://doi.org/10.1109/tpami.2020.3008413>, <http://dx.doi.org/10.1109/TPAMI.2020.3008413>

11. Gehrig, D., Gehrig, M., Hidalgo-Carrio, J., Scaramuzza, D.: Video to events: recycling video datasets for event cameras. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)
12. Gehrig, D., Loquercio, A., Derpanis, K.G., Scaramuzza, D.: End-to-end learning of representations for asynchronous event-based data. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (October 2019)
13. Gerstner, W., Kistler, W.M., Naud, R., Paninski, L.: Neuronal dynamics: from single neurons to networks and models of cognition (2014)
14. Howard, A.G., et al.: MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv 1704.04861 (2017)
15. Huang, G., Liu, Z., Weinberger, K.Q.: Densely connected convolutional networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
16. Kheradpisheh, S.R., Ganjtabesh, M., Thorpe, S.J., Masquelier, T.: STDP-based spiking deep convolutional neural networks for object recognition. *Neural Netw.* **99**, 56–67 (2018)
17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR abs/1412.6980 (2015)
18. Kugele, A., Pfeil, T., Pfeiffer, M., Chicca, E.: Efficient processing of spatio-temporal data streams with spiking neural networks. *Front. Neurosci.* **14**, 439 (2020). <https://doi.org/10.3389/fnins.2020.00439>
19. Lagorce, X., Orchard, G., Galluppi, F., Shi, B.E., Benosman, R.B.: HOTS: a hierarchy of event-based time-surfaces for pattern recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **39**(7), 1346–1359 (2017). <https://doi.org/10.1109/TPAMI.2016.2574707>
20. Lee, C., Kosta, A.K., Zhu, A.Z., Chaney, K., Daniilidis, K., Roy, K.: Spike-FlowNet: event-based optical flow estimation with energy-efficient hybrid neural networks. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12374, pp. 366–382. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-58526-6\\_22](https://doi.org/10.1007/978-3-030-58526-6_22)
21. Lee, J.H., Delbruck, T., Pfeiffer, M.: Training deep spiking neural networks using backpropagation. *Front. Neurosci.* **10**, 508 (2016). <https://doi.org/10.3389/fnins.2016.00508>
22. Lichtsteiner, P., Posch, C., Delbruck, T.: A 128×128 120 dB 15μs latency asynchronous temporal contrast vision sensor. *IEEE J. Solid-State Circuits* **43**(2), 566–576 (2008). <https://doi.org/10.1109/JSSC.2007.914337>
23. Linsley, D., Karkada Ashok, A., Govindarajan, L.N., Liu, R., Serre, T.: Stable and expressive recurrent vision models. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 10456–10467. Curran Associates, Inc. (2020). <https://proceedings.neurips.cc/paper/2020/file/766d856ef1a6b02f93d894415e6bfa0e-Paper.pdf>
24. Liu, S.C., Delbruck, T.: Neuromorphic sensory systems. *Curr. Opin. Neurobiol.* **20**(3), 288–295 (2010)
25. Liu, W., et al.: SSD: single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-46448-0\\_2](https://doi.org/10.1007/978-3-319-46448-0_2)
26. Maro, J.M., Ieng, S.H., Benosman, R.: Event-based gesture recognition with dynamic background suppression using smartphone computational capabilities. *Front. Neurosci.* **14**, 275 (2020)
27. Merolla, P.A., et al.: A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* **345**(6197), 668–673 (2014)

28. Mueggler, E., Rebecq, H., Gallego, G., Delbruck, T., Scaramuzza, D.: The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM **36**, 142–149 (2017)
29. Neftci, E.O., Mostafa, H., Zenke, F.: Surrogate gradient learning in spiking neural networks: bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Process. Mag.* **36**(6), 51–63 (2019)
30. Opez-Vázquez, G., et al.: Evolutionary spiking neural networks for solving supervised classification problems. *Comput. Intell. Neurosci.* **2019**, 13 (2019). <https://doi.org/10.1155/2019/4182639>
31. Orchard, G., Jayawant, A., Cohen, G.K., Thakor, N.: Converting static image datasets to spiking neuromorphic datasets using saccades. *Front. Neurosci.* **9**, 437 (2015). <https://doi.org/10.3389/fnins.2015.00437>
32. Perot, E., De Tournemire, P., Nitti, D., Masci, J., Sironi, A.: Learning to detect objects with a 1 megapixel event camera. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) *Advances in Neural Information Processing Systems*, vol. 33, pp. 16639–16652. Curran Associates, Inc. (2020). <https://proceedings.neurips.cc/paper/2020/file/c213877427b46fa96cff6c39e837ccee-Paper.pdf>
33. Posch, C., Matolin, D., Wohlgenannt, R.: A qVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS. *IEEE J. Solid-State Circuits* **46**(1), 259–275 (2011). <https://doi.org/10.1109/JSSC.2010.2085952>
34. Qiao, N., et al.: A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. *Front. Neurosci.* **9**, 141 (2015). <https://doi.org/10.3389/fnins.2015.00141>
35. Rath, N., Roy, K.: DIET-SNN: Direct input encoding with leakage and threshold optimization in deep spiking neural networks. arXiv 2008.03658 (2020)
36. Rebecq, H., Ranftl, R., Koltun, V., Scaramuzza, D.: Events-to-video: bringing modern computer vision to event cameras. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019)
37. Rieke, F.: *Spikes: Exploring the Neural Code*. MIT Press, Bradford book, Cambridge (1999)
38. Rueckauer, B., Lungu, I.A., Hu, Y., Pfeiffer, M., Liu, S.C.: Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Front. Neurosci.* **11**, 682 (2017). <https://doi.org/10.3389/fnins.2017.00682>
39. Schemmel, J., Brüderle, D., Grünbl, A., Hock, M., Meier, K., Millner, S.: A wafer-scale neuromorphic hardware system for large-scale neural modeling. In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pp. 1947–1950 (2010)
40. Sengupta, A., Ye, Y., Wang, R., Liu, C., Roy, K.: Going deeper in spiking neural networks: VGG and residual architectures. *Front. Neurosci.* **13**, 95 (2019). <https://doi.org/10.3389/fnins.2019.00095>
41. Serrano-Gotarredona, T., Linares-Barranco, B.: A  $128 \times 128$  1.5% contrast sensitivity 0.9% FPN  $3 \mu\text{s}$  latency 4 mW asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers. *IEEE J. Solid-State Circuits* **48**(3), 827–838 (2013). <https://doi.org/10.1109/JSSC.2012.2230553>
42. Shrestha, S.B., Orchard, G.: SLAYER: Spike layer error reassignment in time. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) *Advances in Neural Information Processing Systems*, vol. 31, pp. 1412–1421. Curran Associates, Inc. (2018). <http://papers.nips.cc/paper/7415-slayer-spike-layer-error-reassignment-in-time.pdf>

43. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv 1409.1556 (2015)
44. Sironi, A., Brambilla, M., Bourdis, N., Lagorce, X., Benosman, R.: HATS: histograms of averaged time surfaces for robust event-based object classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
45. Stimberg, M., Brette, R., Goodman, D.F.: Brian 2, an intuitive and efficient neural simulator. *eLife* **8**, e47314 (2019). <https://doi.org/10.7554/eLife.47314>
46. Vidal, A.R., Rebecq, H., Horstschaefer, T., Scaramuzza, D.: Ultimate SLAM? Combining events, images, and IMU for robust visual SLAM in HDR and high-speed scenarios. *IEEE Robot. Autom. Lett.* **3**(2), 994–1001 (2018). <https://doi.org/10.1109/LRA.2018.2793357>
47. Wu, Y., Deng, L., Li, G., Zhu, J., Xie, Y., Shi, L.: Direct training of spiking neural networks: faster, larger, better. In: Proceedings of the AAAI Conference on Artificial Intelligence (2019)
48. Zhu, D., et al.: Neuromorphic visual odometry system for intelligent vehicle application with bio-inspired vision sensor. In: 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 2225–2232. IEEE (2019)