Recurrent Vision Transformers for Object Detection with Event Cameras

Mathias Gehrig and Davide Scaramuzza
Robotics and Perception Group, University of Zurich, Switzerland

Abstract

We present Recurrent Vision Transformers (RVTs), a novel backbone for object detection with event cameras. Event cameras provide visual information with sub-millisecond latency at a high-dynamic range and with strong robustness against motion blur. These unique properties offer great potential for low-latency object detection and tracking in time-critical scenarios. Prior work in event-based vision has achieved outstanding detection performance but at the cost of substantial inference time, typically beyond 40 milliseconds. By revisiting the high-level design of recurrent vision backbones, we reduce inference time by a factor of 6 while retaining similar performance. To achieve this, we explore a multi-stage design that utilizes three key concepts in each stage: first, a convolutional prior that can be regarded as a conditional positional embedding. Second, local and dilated global self-attention for spatial feature interaction. Third, recurrent temporal feature aggregation to minimize latency while retaining temporal information. RVTs can be trained from scratch to reach state-of-the-art performance on event-based object detection - achieving an mAP of 47.2% on the Gen1 automotive dataset. At the same time, RVTs offer fast inference ($<$ 12 ms on a T4 GPU) and favorable parameter efficiency (5× fewer than prior art). Our study brings new insights into effective design choices that can be fruitful for research beyond event-based vision.

Code: https://github.com/uzh-rpg/RVT

1. Introduction

Time matters for object detection. In 30 milliseconds, a human can run 0.3 meters, a car on public roads covers up to 1 meter, and a train can travel over 2 meters. Yet, during this time, an ordinary camera captures only a single frame.

Frame-based sensors must strike a balance between latency and bandwidth. Given a fixed bandwidth, a frame-based camera must trade-off camera resolution and frame rate. However, in highly dynamic scenes, reducing the resolution or the frame rate may come at the cost of missing essential scene details, and, in safety-critical scenarios like automotive, this may even cause fatalities.

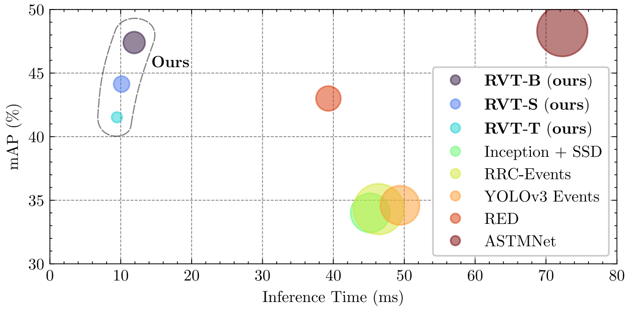


Figure 1. **Detection performance vs inference time** of our RVT models on the 1 Mpx detection dataset using a T4 GPU. The circle areas are proportional to the model size.

In recent years, event cameras have emerged as alternative sensor that offers a different trade-off. Instead of counterbalancing bandwidth requirements and perceptual latency, they provide visual information at sub-millisecond latency but sacrifice absolute intensity information. Instead of capturing intensity images, event cameras measure changes in intensity at the time they occur. This results in a stream of events, which encode time, location, and polarity of brightness changes [14]. The main advantages of event cameras are their sub-millisecond latency, very high dynamic range ($>$ 120 dB), strong robustness to motion blur, and ability to provide events asynchronously in a continuous manner.

In this work, we aim to utilize these outstanding properties of event cameras for object detection in time-critical scenarios. Therefore, our objective is to design an approach that reduces the processing latency as much as possible while maintaining high performance. This is challenging because event cameras asynchronously trigger binary events that are spread of pixel space and time. Hence, we need to develop detection algorithms that can continuously associate features in the spatio-temporal domain while simultaneously satisfying strict latency requirements.

Recent work has shown that dynamic graph neural networks (GNNs) [28, 43] and sparse neural networks [10, 34, 55, 57] can theoretically achieve low latency inference for event-based object detection. Yet, to achieve this in practical scenarios they either require specialized hardware or their detection performance needs to be improved.

An alternative thread of research approaches the problem from the view of conventional, dense neural network designs [7,19,20,26,38]. These methods show impressive performance on event-based object detection, especially when using temporal recurrence in their architectures [26, 38]. Still, the processing latency of these approaches remains beyond 40 milliseconds such that the low-latency aspect of event cameras cannot be fully leveraged. This raises the question: How can we achieve both high accuracy and efficiency without requiring specialized hardware?

We notice that common design choices yield a suboptimal trade-off between performance and compute. For example, prior work uses expensive convolutional LSTM (Conv-LSTM) cells [44] extensively in their feature extraction stage [26, 38] or relies on heavy backbones such as the VGG architecture [26]. Sparse neural networks instead struggle to model global mixing of features which is crucial to correctly locate and classify large objects in the scene.

To achieve our main objective, we fundamentally revisit the design of vision backbones for event-based object detection. In particular, we take inspiration from neural network design for conventional frame-based object detection and combine them with ideas that have proven successful in the event-based vision literature. Our study deliberately focuses on *macro* design of the object detection backbone to identify key components for both high performance and fast inference on GPUs. The resulting neural network is based on a single block that is repeated four times to form a multi-stage hierarchical backbone that can be used with off-the-shelf detection frameworks.

We identify three key components that enable an excellent trade-off between detection performance and inference time. First, we find that interleaved local- and global self-attention [50] is ideally suited to mix both local and global features while offering linear complexity in the input resolution. Second, this attention mechanism is most effective when preceded by a simple convolution that also downsamples the spatial resolution from the previous stage. This convolution effectively provides a strong prior about the grid-structure of the pixel array and also acts as a conditional positional embedding for the transformer layers [9]. Third, temporal recurrence is paramount to achieve strong detection performance with events. Differently from prior work, we find that Conv-LSTM cells can be replaced by plain LSTM cells [18] that operate on each feature separately[1]. By doing so, we dramatically reduce the number of parameters and latency but also slightly improve the overall performance. Our full framework achieves competitive performance and higher efficiency compared to state-of-the-art methods. Specifically, we reduce parameter count (from 100M to 18.5 M) and inference time (from 72 ms to 12 ms) up to a factor of 6 compared to prior art [26]. At the same time, we train our networks from scratch, showing that these benefits do not originate from large-scale pretraining.

Our paper can be summarized as follows: (1) We reexamine predominant design choices in event-based object detection pipelines and reveal a set of key enablers for high performance in event-based object detection. (2) We propose a simple, composable stage design that unifies the crucial building blocks in a compact way. We build a 4-stage hierarchical backbone that is fast, lightweight and still offers performance comparable to the best reported so far. (3) We achieve state-of-the-art object detection performance of 47.2% mAP on the Gen1 detection dataset [11] and a highly competitive 47.4% mAP on the 1 Mpx detection dataset [38] while training the proposed architecture from scratch. In addition, we also provide insights into effective data augmentation techniques that contribute to these results.

2. Related Work

2.1. Object Detection for Event Cameras

Object detection in the event camera literature can be broadly classified into three emerging research directions.

Recent work explores **graph neural networks** to dynamically construct a spatio-temporal graph [28, 35, 43]. New nodes and node edges are established by sub-sampling events and finding existing nodes that are close in space-time. The main challenge is to design the architecture such that information can propagate over vast distances in the space-time volume. This is relevant, for example, when large objects move slowly with respect to the camera. Furthermore, aggressive sub-sampling of events can lead to the removal of potentially crucial information, but is often required to maintain low-latency inference.

A second line of work employs **spiking neural networks** (SNNs) that propagate information sparsely within the network [10, 55, 57]. SNNs are closely related to dense recurrent neural networks (RNNs) in that each spiking neuron has an internal state that is propagated in time. Differently from RNNs, neurons in SNNs only emit spikes whenever a threshold is reached. This spike generation mechanism is not differentiable, which leads to substantial difficulties in optimizing these networks [4,22,23,36,45,47,56]. One workaround is to avoid the aforementioned threshold and instead propagate features throughout the receptive field [34] . The downside of this mechanism is that the sparse-processing property is lost within deeper layers of the network. Overall, the design and training of SNNs still requires fundamental investigation before competitive performance can be reached.

A third research direction is concerned with exploring

[1] equivalent to 1×1 kernel in a Conv-LSTM cell

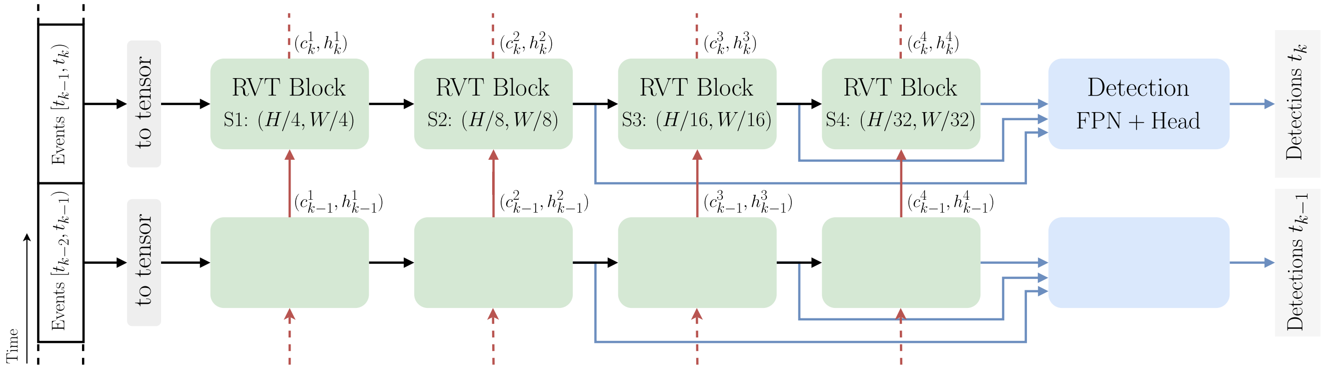


Figure 2. **Overview** of the unrolled computation graph of our multi-stage recurrent backbone. Events are processed into a tensor representation before they are used as input to the first stage. Each stage also reuses the LSTM states (c: cell, h: hidden) from the previous timestep. Finally, the detection framework interfaces with the backbone from the second stage onwards. Specifically, the hidden states of the LSTMs are used as features for the detection framework.

dense neural networks for object detection with event cameras. The first step is the creation of a dense tensor (event representation) that enables compatibility with dense operations such as convolutions. Early work directly uses a single event representation generated from a short temporal window of events to infer detections [7, 19, 20]. These approaches discard relevant information from beyond the considered temporal window such that detecting slowly moving objects becomes difficult or impossible. Followup work addresses this issue by incorporating recurrent neural network layers [26,38] which drastically improved the detection performance. We follow this line of work but revamp dominant architecture choices to build a canonical framework that is fast, lightweight and highly performant.

2.2. Vision Transformers for Spatio-Temporal Data

The success of attention-based models [52] in NLP has inspired the exploration of transformer-based architectures in computer vision [12]. Attention-based models have recently also been explored in video classification [1,5,13,48] where the models are applied directly to a set of frames. While these approaches have shown promising results in spatio-temporal modelling, they are optimized for offline processing of stored video data.

In event-based vision, attention-based components have found applications in classification [42, 53] and image reconstruction [54], and monocular depth estimation [31], but their use in object detection has yet to be investigated.

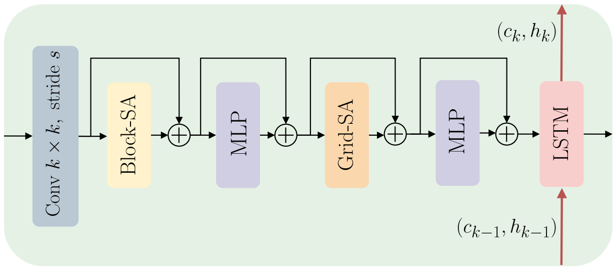


Figure 3. **RVT block structure**. The input is convolved with kernel size $k \times k$ and stride s. Block-SA applies self-attention in local windows while Grid-SA is a global operation using dilated attention. Finally, each block ends with an LSTM that reuses the (cell- and hidden) states from the previous timestep. The LSTM is applied to each feature separately. Normalization and activation layers are omitted for conciseness.

3. Method

Our object detection approach is designed to process a stream of events sequentially as they arrive. Incoming events are first processed into tensors that represent events in space and time. In every timestep, our network takes a new event representation as input as well as the previous states of the recurrent neural network layers. After each pass through the backbone, the output of the RNNs are used as input to the detection framework. The following sections elaborate on each one of these steps. Fig. 2 shows an overview of the RVT architecture.

Event Processing Each pixel of an event camera can independently trigger an event when a significant log brightness change occurs. An event can be positive or negative depending on the sign of the brightness change. We characterize an event with polarity $p_k \in \{0, 1\}$ as a tuple $e_k = (x_k, y_k, t_k, p_k)$ that occurs at pixel (x_k, y_k) at time t_k. Modern event cameras can produce 10s of millions of events per second which renders event-by-event processing out of reach on conventional processing units.

In this work we opt for a very simple preprocessing step to enable compatibility with convolutional neural network layers which, as we will show later in Sec. 4.2, are an important contributor to the performance of our model.

Our preprocessing step starts with the creation of a 4-dimensional tensor E. The first dimension consists of two components and represents the polarity. The second dimension has T components and is associated with T discretization steps of time. The 3rd and 4th dimension represent height and width of the event camera. We process set of events $\mathcal{E}$ within a time duration $[t_a, t_b)$ the following way:

$$E(p, \tau, x, y) = \sum_{e_k \in \mathcal{E}} \delta(p - p_k)\delta(x - x_k, y - y_k)\delta(\tau - \tau_k),$$

$$\tau_k = \left\lfloor \frac{t_k - t_a}{t_b - t_a} \cdot T \right\rfloor$$

In words, we create T 2-channel frames where each pixel contains the number of positive or negative events within one of the T temporal frames. As a final step, we flatten the polarity and time dimension to retrieve a 3-dimensional tensor with shape $(2T, H, W)$ to directly enable compatibility with 2D convolutions. We implement the presented algorithm with byte tensors to save memory and bandwidth. Other, more sophisticated representations are possible [2, 3, 6, 16, 26, 51, 58], but their thorough evaluation is not our focus.

3.1. Mixing Spatial and Temporal Features

The main difficulty of object detection with event cameras is that at any given time, the neural network should be able to efficiently (1) extract local- and global task-relevant features in pixel space because objects can cover both very small regions or large portions of the field of view; (2) extract features from very recent events (e.g. moving edges) as well as events from several seconds ago. This is necessary because some objects are moving slowly with respect to the camera such that they generate very few events over time. These observations motivate us to investigate transformer layers for spatial feature extraction and recurrent neural networks for efficient temporal feature extraction. Fig. 3 illustrates the components of a single stage.

Spatial Feature Extraction The spatial feature extraction stage should incorporate a prior about the fact that pixels are arranged in a 2D grid as early as possible in the computation graph. We enable this by using a convolution with overlapping kernels on the input features that at the same time spatially downsamples the input or features from the previous stage. This convolution also endows our model with a conditional positional embedding [9] such that we do not require absolute [12, 52] or relative [32] positional embeddings. Our ablation study in Sec. 4.2 shows that overlapping kernels lead to a substantial boost in detection performance.

In a subsequent step, the resulting features are transformed through multi-axis self-attention. We quickly summarize the steps but refer to Tu et. al [50] for an elaborate explanation. Multi-axis attention consists of two stages using self-attention. The first stage performs local feature interaction while the second stage enables dilated global feature mixing. More specifically, the features are first grouped locally into non-overlapping windows: Let $X \in \mathbb{R}^{H \times W \times C}$ be the input feature tensor. We reshape the tensor to a shape $(\frac{H}{P} \times \frac{W}{P}, P \times P, C)$ where $P \times P$ is the window size in which multi-head self-attention [52] is applied. This *block attention* (Block-SA in Fig. 3) is used to model local interactions. As a next step, we would ideally be able to extract features globally. One straightforward way to achieve this would be applying self-attention on the whole feature map. Unfortunately, global self-attention has quadratic complexity in the number of features. Instead, we use *grid attention* (Grid-SA in Fig. 3). Grid attention partitions the feature maps into a grid of shape $(G \times G, \frac{H}{G} \times \frac{W}{G}, C)$ using a $G \times G$ uniform grid. The resulting windows are of size $\frac{H}{G} \times \frac{W}{G}$. Self-attention is then applied to these windows which corresponds to global, dilated mixing of features.

We study alternative designs as part of our architecture in the ablation studies in Sec. 4.2.

Temporal Feature Extraction We opt for temporal feature aggregation with LSTM [18] cells at the end of the stage. Differently from prior work [26, 38] we find that temporal and spatial feature aggregation can be completely separated. This means that we use plain LSTM cells such that the states of the LSTMs do not interact with each other. By avoiding Conv-LSTM units [44], we can drastically reduce the computational complexity and parameter count. I.e. a Conv-LSTM with kernel size $k \times k$ and stride 1 demands k^2 the number of parameters and compute compared to the original LSTM cell. We examine this aspect in the experimental Sec. 4.2.

Model Details We apply LayerNorm [24] before and LayerScale [49] after each attention and MLP module, and add a residual connection after each module. We found that LayerScale enables a wider range of learning rates.

3.2. Hierarchical Multi-Stage Design

We compose multiple RVT blocks together to form a multi-stage hierarchical backbone. The overall architecture is shown in Fig. 2.

At first, a local temporal slice of events is processed into a 2D tensor format as formulated in the beginning of this section. Subsequently, each stage takes the previous features as input and optionally uses the LSTM state from the last timestep to compute features for the next stage. By saving the LSTM states for the following timestep, each re-

				Channels		
Stage	Size	Kernel	Stride	RVT-T	RVT-S	RVT-B
S1	1/4	7	4	32	48	64
S2	1/8	3	2	64	96	128
S3	1/16	3	2	128	192	256
S4	1/32	3	2	256	384	512

Table 1. **RVT parameters and architecture variation**. All model variants share the same parameter set except the number of channels per stage. Each stage initially applies a 2D convolution with kernel and stride as indicated in the table.

current stage can retain temporal information for the whole feature map.

We follow prior work and use features from the second to the fourth stage for the object detection framework. To do so, we reshape the hidden states of the LSTMs into 2D feature maps.

4. Experiments

We conduct ablations and evaluation our model on the Gen1 [11] and 1 Mpx [38] event camera datasets. We train three variants of our model on both datasets: the base model RVT-B and its small and tiny variants RVT-S and RVT-T. Parameter details for the models are shown in Tab. 1.

4.1. Setup

Implementation Details We initialize all layers randomly except LayerScale which is initialized to 1e-5 for each module. Our models are trained with mixed precision for 400k iterations with the ADAM optimizer [21] using a OneCycle learning rate schedule [46] with a linear decay from a maximum learning rate. We use a mixed batching strategy that applies backpropagation through time (BPTT) to half of the samples of the batch and truncated BPTT (TBPTT) to the other half. More details regarding this batching strategy can be found in the supplementary material. Our data augmentation includes random horizontal flipping, zooming in and zooming out. More details on data augmentation are available in Sec. 4.2 and the supplementary material. To construct event representations, we consider 50 ms time windows that are discretized into $T = 10$ bins. Finally, we use the YOLOX framework [15], which includes the IOU loss, class loss and regression loss. These losses are averaged both over the batch and sequence length for each optimization step.

To compare against prior work on the Gen1 dataset, we train our models with a batch size of 8, sequence length of 21, and learning rate of 2e-4. The training takes approximately 2 days on a single A100 GPU.

On the 1 Mpx dataset, we train with a batch size of 24, sequence length of 5, and learning rate of 3.5e-4. The training takes approximately 3 days on two A100 GPUs.

	Gen1		1 Mpx		
Block-type	mAP	AP_{50}	mAP	AP_{50}	Params (M)
multi-axis	**47.6**	**70.1**	**46.0**	**72.3**	18.5
Swin	46.7	68.7	44.4	71.7	18.5
ConvNeXt	45.5	65.8	42.3	70.6	18.7

Table 2. **Spatial Aggregation**. Multi-axis attention leads to the best results on both the Gen1 and 1 Mpx dataset.

Datasets The Gen1 Automotive Detection dataset [11] consists of 39 hours of event camera recordings at a resolution of 304 × 240. In total, the Gen1 dataset contains 228k car and 28k pedestrian bounding boxes available at 1, 2 or 4 Hz. We follow the evaluation protocol of prior work [26,38] and remove bounding boxes with a side length of less than 10 pixels and a diagonal of less than 30 pixels.

The 1 MPx dataset [38] also features driving scenarios but provides recordings at a higher resolution of 720 × 1280 over a period of several months at day and night. It consists of approximately 15 hours of event data labeled at a frequency of 30 or 60 Hz with a total amount of 25 million bounding box labels for three classes (car, pedestrian, and two-wheeler). We follow the evaluation protocol of prior work [26, 38]. That is, we remove bounding boxes with a side length of less than 20 pixels and a diagonal of less than 60 pixels and halve the input resolution to nHD resolution (640 × 360). We provide qualitative examples of this dataset together with predictions of our base model in Fig. 4.

For both datasets, mean average precision (mAP) is the main metric [29] that we consider.

4.2. Ablation Studies

This section examines the two main contributors to the final performance of the proposed model. First, we investigate key components and design choices of the proposed backbone. Second, we study the influence of different data augmentation techniques that are compatible with our sequential problem setting.

Unless stated otherwise, the ablation studies are performed on the Gen1 validation set using the best performing model after 400k iterations. To reduce the training time, we use BPTT with a sequence length of 11 instead 21.

4.2.1 Model Components

Spatial Interaction In Tab. 2, we study different spatial aggregation techniques. For a fair comparison, we keep the LSTM and convolutional downsampling layers identical and only exchange the attention and MLP modules. We

Conv. kernel type	mAP	AP_{50}	AP_{75}	Params (M)
overlapping	**47.6**	**70.1**	**52.6**	18.5
non-overlapping	46.1	68.6	50.5	**17.6**

Table 3. **Downsampling Strategy**. The usage of overlapping kernels leads to higher performance at the expense of a slight increase in the number of parameters.

LSTM kernel size	mAP	AP_{50}	AP_{75}	Params (M)
1×1	**47.6**	**70.1**	**52.6**	**18.5**
3×3	46.5	69.0	51.4	40.8
3×3 depth-sep	46.3	67.2	51.2	18.6

Table 4. **LSTM kernel size**. Conv-LSTM variants do not outperform the feature specific (1×1) LSTM.

S1	S2	S3	S4	mAP	AP_{50}	AP_{75}
				32.0	54.8	31.4
			✓	39.8	63.5	41.6
		✓	✓	44.2	68.4	47.5
	✓	✓	✓	46.9	70.0	50.8
✓	✓	✓	✓	**47.6**	**70.1**	**52.6**

Table 5. **LSTM placement**. LSTM cells contribute to the overall performance even in the early stages.

compare multi-axis attention with ConvNext blocks [33] and Swin transformer blocks [32]. ConvNext is a convolutional neural network architecture that has shown competitive performance with transformer-based models on a wide range of tasks, including object detection. We use the default kernel size of 7 × 7 as originally suggested and place three ConvNeXt blocks in each stage to approximately match the number of parameters of the reference model. Swin, instead, is an attention-based model that applies local self-attention in windows that interact with each other through cyclic shifting.

We find that our Swin variant achieves better performance than the ConvNext variant, however, both are outperformed by multi-axis self-attention [50] on both the Gen1 and 1 Mpx dataset. This experiment suggests that global interaction at every stage (multi-axis) is advantageous to purely local interaction (Swin, ConvNext).

Convolutional Downsampling The original vision transformer [12] architecture does not perform local feature interaction with convolutional layers. Some popular hierarchical counterparts also choose to apply downsample features without overlapping kernels [8, 32]. In Tab. 3, we compare overlapping and non-overlapping convolutional kernels in both the input layer (patch embedding) and feature downsampling stage. While non-overlapping convolutions reduce the number of parameters, they cause a substantial drop in performance. Consequently, we choose overlapping kernels in all stages of the network.

LSTM with Convolutions Prior state-of-the-art approaches on object detection with event cameras heavily rely on convolutional LSTM cells [26, 38]. We revisit this design choice and experiment with plain LSTM cells and a depthwise separable Conv-LSTM variant [39]. The depthwise separable Conv-LSTM first applies a depthwise separable convolution on both the input and hidden state before a point-wise (1×1) convolution is applied. Our results in Tab. 4 suggest that plain LSTM cells are sufficient in our model and even outperform both variations. This is to some degree surprising because both variants are a strict superset of the plain LSTM. We decide to use a plain LSTM cell based on these observations.

LSTM Placement In this ablation we study the influence of using temporal recurrence only in a subset of stages or not at all. For all comparisons, we leave the model exactly the same but reset the states of the LSTMs at selected stages in each timestep. This way, we can simulate the absence of recurrent layers while keeping the number of parameters constant in the comparisons.

The results in Tab. 5 suggest that using no recurrence at all leads to a drastic decline of detection performance. Enabling the LSTMs in each stage, starting from the fourth consistently leads to enhanced performance. Surprisingly, we find that adding an LSTM to the first stage also leads to improvements, albeit the increase in mAP is not large. In general, this experiment suggests that the detection framework benefits from features that have been augmented with temporal information. Based on our observations, we decide to keep the LSTM also in the first stage.

4.2.2 Data Augmentation

While data augmentation is not directly related to the model itself, it greatly influences the final result as we will illustrate next. Here, we investigate three data augmentation techniques that are suitable for object detection on spatio-temporal data: Random (1) horizontal flipping, (2) zoom-in, and (3) zoom-out.

Zoom-in augmentation randomly selects crops that contain at least one full bounding box at the final timestep of the BPTT sequence (i.e. during training). This crop is then applied to the rest of the sequence before the crops are rescaled to the default resolution. This procedure ensures that we have at least a single label to compute the loss function while maintaining the same resolution during training.

			Gen1		1 Mpx		
Method	Backbone	Detection Head	mAP	Time (ms)	mAP	Time (ms)	Params (M)
NVS-S [27]	GNN	YOLOv1 [40]	8.6	-	-	-	0.9
Asynet [34]	Sparse CNN	YOLOv1	14.5	-	-	-	11.4
AEGNN [43]	GNN	YOLOv1	16.3	-	-	-	20.0
Spiking DenseNet [10]	SNN	SSD [30]	18.9	-	-	-	8.2
Inception + SSD [19]	CNN	SSD	30.1	19.4	34.0	45.2	> 60*
RRC-Events [7]	CNN	YOLOv3 [41]	30.7	21.5	34.3	46.4	> 100*
MatrixLSTM [6]	RNN + CNN	YOLOv3	31.0	-	-	-	61.5
YOLOv3 Events [20]	CNN	YOLOv3	31.2	22.3	34.6	49.4	> 60*
RED [38]	CNN + RNN	SSD	40.0	16.7	43.0	39.3	24.1
ASTMNet [26]	(T)CNN + RNN	SSD	46.7	35.6	**48.3**	72.3	> 100*
RVT-B (ours)	Transformer + RNN	YOLOX [15]	**47.2**	**10.2 (3.7)**	47.4	**11.9 (6.1)**	18.5
RVT-S (ours)	Transformer + RNN	YOLOX	46.5	9.5 (3.0)	44.1	10.1 (5.0)	9.9
RVT-T (ours)	Transformer + RNN	YOLOX	44.1	9.4 (2.3)	41.5	9.5 (3.5)	4.4

Table 6. **Comparisons on test sets of Gen1 and 1 Mpx datasets**. Best results in **bold** and second best underlined. Brackets $(\cdot)$ in runtime indicate the inference time with JIT-compiled code using `torch.compile`. A star * suggests that this information was not directly available and estimated based on the publications. Runtime is measured in milliseconds for a batch size of 1. We used a T4 GPU for RVT to compare against indicated timings in prior work [26, 38] on comparable GPUs (Titan Xp).

h-flip	zoom-in	zoom-out	mAP	AP_{50}	AP_{75}
			38.1	59.5	41.1
✓			41.6	63.5	45.5
	✓		45.8	67.8	49.8
		✓	44.1	65.7	48.4
✓	✓	✓	**47.6**	**70.1**	**52.6**

Table 7. **Data Augmentation**. Data augmentation consistently improves the results.

Zoom-out augmentation resizes the full input to a lower resolution and randomly places the downscaled input in a zero-tensor initialized at the default resolution. This procedure is then applied in an identical way to the remaining BPTT sequence.

Table 7 shows that our model is performing poorly if no data augmentation is applied. Overall, we find that data augmentation is important to combat overfittig not only on the Gen1 sequence but also on the 1 Mpx dataset. The most effective augmentation is zoom-in, followed by zoom-out and horizontal flipping. Based on these results, we decide to apply all data augmentation techniques. We report the specific hyperparameters in the supplementary material.

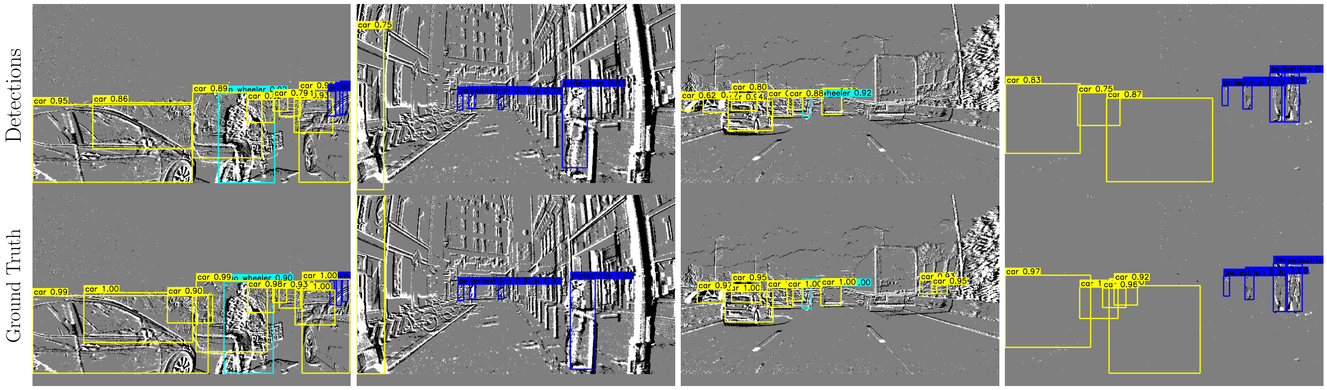
4.3. Benchmark Comparisons

In this section, we compare our proposed neural network architecture against prior work on both the Gen1 [11] and 1 Mpx dataset [38] and summarize the results in Tab. 6. We train three models, a base model (RVT-B) with approximately 18.5 million parameters, a small variant (RVT-S) with 9.9 million parameters, and a tiny model (RVT-T) with 4.4 million parameters by adapting the channel dimensions in each stage. Their architectural hyperparameters are outlined in Tab. 1. To compare with prior work, we choose the models based on their best performance on the validation set and evaluate them on the test set.

From Tab. 6 we can draw multiple conclusions. First, we observe that models using recurrent layers consistently outperform other approaches, both sparse (GNNs, SNNs) and dense feed-forward models without recurrent layers (Inception+SSD, RRC-Events, YOLOv3 Events) by an mAP of more than 10 on both datasets. One notable exception is MatrixLSTM [6] which applies LSTM cells directly at the input. In contrast, RED [38] and ASTMNet [26] employ recurrent layers only in deeper layers.

Our base model achieves a new state-of-the-art performance of 47.2 mAP on the Gen1 dataset and 47.4 mAP on the 1 Mpx dataset. ASTMNet claims comparable results on both datasets albeit at the cost of using a much larger backbone and increased inference time. The RED model, also reports favorable results, but achieves 7.2 lower mAP on the Gen1 and 4.4 lower mAP on the 1 Mpx dataset compared to our model. Finally, our tiny model is amongst the smallest in our comparison. Still, it achieves 4.1 higher mAP on the Gen1 dataset than the RED model while using 5 times fewer parameters.

Inference Time We also compute the inference time of our model on a T4 GPU with a batch size of 1. Unfortu-



(a) (b) (c) (d)

Figure 4. **Predictions on the 1 Mpx dataset**. All examples are thematically picked to illustrate the behaviour of the model in different scenarios. (d) shows a scenario in which the model can still partially detect objects in absence of events due the temporal memory.

nately, both RED and ASTMNet are not open source such that we cannot directly compare inference time on the same GPU model. Instead, we use the timings provided by the authors that conducted their timing experiments on comparable GPUs (e.g. Titan Xp). We report the timing results of our models in Tab. 6 and also visualize them in Fig. 1.

To compare against prior work we first compute the inference time in PyTorch eager mode. In eager mode, our base model achieves an inference time of 10.2 milliseconds (ms) on the Gen1 dataset (304 × 240 resolution). This implies a latency reduction of 6 ms compared to RED and over 3 times lower inference time than ASTMNet. On the the 1 Mpx dataset, at a resolution of 640 × 360, our base model takes 11.9 ms for a forward pass, which is 3 times faster than RED and over 5 times faster than ASTMNet.

Even on a T4 GPU, most of the inference time is framework overhead. To overcome this partially, we use the JIT compilation feature `torch.compile` of PyTorch 2 [37]. As Tab. 6 shows, this almost halves the inference time for RVT-B on the 1 Mpx dataset and reduces the inference time by a factor of 2.7 on the Gen1 dataset. As expected, the small and tiny models benefit even more from JIT compilation. For example, RVT-T only takes 2.3 ms for a forward pass on Gen1 and 3.5 ms on 1 Mpx. On a RTX 3090 GPU, RVT-B completes a forward pass in 2.8 ms on the 1 Mpx dataset, which shows the potential for low-latency inference if power consumption is less of a concern.

5. Discussion and Limitations

We use a very simple event representation which does not leverage the full potential of event-based data. For example, we only have a weak prior on the order of events because we process the temporal dimension directly with fully connected layers. Recent work has shown substantial gains by introducing temporal convolutions in early layers [26]. Efficient low-level processing of event data is still an open research problem that we have not addressed in this work.

Our approach currently only uses event streams to detect objects. Frames yield complementary information that, when properly incorporated, will yield significantly enhanced detection performance. For example, in Fig. 4 (d) we can see that our model can retain information over some period when no events are available. Still, the memory of the network will fade and detection performance deteriorates. High quality frames even at low frame-rate could provide the missing complementary information. Hence, we believe that a multi-modal extension of our method on a suitable dataset [17, 25] is a promising next step.

6. Conclusion

We introduced a novel backbone architecture for object detection with event cameras. The architecture consists of a stage design that is repeatedly applied to create a multi-stage hierarchical neural network. Each stage compactly incorporates convolutional priors, local- and sparse global attention and recurrent feature aggregation. Our experiments highlight that recurrent vision transformers can be trained from scratch to reach state-of-the-art performance in object detection with event cameras. The resulting canonical stage-design is directly compatible with existing detection frameworks, and paves the way to low-latency object detection with event cameras on conventional hardware. Nonetheless, we hope that this work also inspires novel designs in future neuromorphic systems.

7. Acknowledgment

This work was supported by the National Centre of Competence in Research (NCCR) Robotics (grant agreement No. 51NF40-185543) through the Swiss National Science Foundation (SNSF), and the European Research Council (ERC) under grant agreement No. 864042 (AGILEFLIGHT).

References

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Int. Conf. Comput. Vis. (ICCV)*, 2021.

[2] Raymond Baldwin, Ruixu Liu, Mohammed Mutlaq Almatrafi, Vijayan K Asari, and Keigo Hirakawa. Time-ordered recent event (TORE) volumes for event cameras. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2022.

[3] Sami Barchid, Jose Mennesson, and Chaabane Djeraba. Bina-rep event frames: A simple and effective representation for event-based cameras. In *IEEE Int. Conf. Image Process. (ICIP)*, 2022.

[4] Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications*, 2020.

[5] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *Proc. Int. Conf. Mach. Learning (ICML)*, 2021.

[6] Marco Cannici, Marco Ciccone, Andrea Romanoni, and Matteo Matteucci. Asynchronous convolutional networks for object detection in neuromorphic cameras. In *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*, 2019.

[7] Nicholas F. Y. Chen. Pseudo-labels for supervised learning on dynamic vision sensor data, applied to object detection under ego-motion. In *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*, 2018.

[8] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. In *Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2021.

[9] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Conditional positional encodings for vision transformers, 2021.

[10] Loc Cordone, Benot Miramond, and Philippe Thierion. Object detection with spiking neural networks on automotive event data. In *Int. Joint Conf. Neural Netw. (IJCNN)*, 2022.

[11] Pierre de Tournemire, Davide Nitti, Etienne Perot, Davide Migliore, and Amos Sironi. A large scale event-based detection dataset for automotive, 2020.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Int. Conf. Learn. Representations (ICLR)*, 2021.

[13] Christoph Feichtenhofer, Haoqi Fan, Yanghao Li, and Kaiming He. Masked autoencoders as spatiotemporal learners. *arXiv e-prints*, 2022.

[14] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Jörg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based vision: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020.

[15] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021, 2021.

[16] Daniel Gehrig, Antonio Loquercio, Konstantinos G. Derpanis, and Davide Scaramuzza. End-to-end learning of representations for asynchronous event-based data. In *Int. Conf. Comput. Vis. (ICCV)*, 2019.

[17] Mathias Gehrig, Willem Aarents, Daniel Gehrig, and Davide Scaramuzza. Dsec: A stereo event camera dataset for driving scenarios. *IEEE Robot. Autom. Lett.*, 2021.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[19] Massimiliano Iacono, Stefan Weber, Arren Glover, and Chiara Bartolozzi. Towards event-driven object detection with off-the-shelf deep learning. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2018.

[20] Zhuangyi Jiang, Pengfei Xia, Kai Huang, Walter Stechele, Guang Chen, Zhenshan Bing, and Alois Knoll. Mixed frame-/event-driven fast pedestrian detection. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2019.

[21] Diederik P. Kingma and Jimmy L. Ba. Adam: A method for stochastic optimization. *Int. Conf. Learn. Representations (ICLR)*, 2015.

[22] Chankyu Lee, Syed Shakib Sarwar, Priyadarshini Panda, Gopalakrishnan Srinivasan, and Kaushik Roy. Enabling spike-based backpropagation for training deep neural network architectures. *Front. Neurosci.*, 2020.

[23] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Front. Neurosci.*, 2016.

[24] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv e-prints*, 2016.

[25] Jianing Li, Siwei Dong, Zhaofei Yu, Yonghong Tian, and Tiejun Huang. Event-based vision enhanced: A joint detection framework in autonomous driving. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, 2019.

[26] Jianing Li, Jia Li, Lin Zhu, Xijie Xiang, Tiejun Huang, and Yonghong Tian. Asynchronous spatio-temporal memory network for continuous event-based object detection. *IEEE Trans. Image Process.*, 2022.

[27] Yijin Li, Han Zhou, Bangbang Yang, Ye Zhang, Zhaopeng Cui, Hujun Bao, and Guofeng Zhang. Graph-based asynchronous event processing for rapid object recognition. In *Int. Conf. Comput. Vis. (ICCV)*, October 2021.

[28] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2021.

[29] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 740–755. 2014.

[30] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot MultiBox detector. In *Eur. Conf. Comput. Vis. (ECCV)*, 2016.

[31] Xu Liu, Jianing Li, Xiaopeng Fan, and Yonghong Tian. Event-based monocular dense depth estimation with recurrent transformers. *arXiv e-prints*, 2022.

[32] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Int. Conf. Comput. Vis. (ICCV)*, 2021.

[33] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2022.

[34] Nico Messikommer, Daniel Gehrig, Antonio Loquercio, and Davide Scaramuzza. Event-based asynchronous sparse convolutional networks. In *Eur. Conf. Comput. Vis. (ECCV)*, 2020.

[35] Anton Mitrokhin, Zhiyuan Hua, Cornelia Fermuller, and Yiannis Aloimonos. Learning visual motion segmentation using event surfaces. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, June 2020.

[36] Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 2019.

[37] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2019.

[38] Etienne Perot, Pierre de Tournemire, Davide Nitti, Jonathan Masci, and Amos Sironi. Learning to detect objects with a 1 megapixel event camera. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2020.

[39] Andreas Pfeuffer and Klaus Dietmayer. Separable convolutional LSTMs for faster video segmentation. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019.

[40] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2016.

[41] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018.

[42] Alberto Sabater, Luis Montesano, and Ana C. Murillo. Event transformer. a sparse-aware solution for efficient event data processing. In *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*, 2022.

[43] Simon Schaefer, Daniel Gehrig, and Davide Scaramuzza. Aegnn: Asynchronous event-based graph neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2022.

[44] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2015.

[45] Sumit Bam Shrestha and Garrick Orchard. SLAYER: Spike layer error reassignment in time. In *Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2018.

[46] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates, 2017.

[47] Aboozar Taherkhani, Ammar Belatreche, Yuhua Li, Georgina Cosma, Liam P. Maguire, and T.M. McGinnity. A review of learning in biologically plausible spiking neural networks. *Neural Netw.*, 2020.

[48] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. VideoMAE: Masked autoencoders are data-efficient learners for self-supervised video pre-training. In *Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2022.

[49] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Int. Conf. Comput. Vis. (ICCV)*, 2021.

[50] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. MaxViT: Multi-axis vision transformer. In *Eur. Conf. Comput. Vis. (ECCV)*, 2022.

[51] Stepan Tulyakov, Francois Fleuret, Martin Kiefel, Peter Gehler, and Michael Hirsch. Learning an event sequence embedding for dense event-based deep stereo. In *Int. Conf. Comput. Vis. (ICCV)*, October 2019.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2017.

[53] Zuowen Wang, Yuhuang Hu, and Shih-Chii Liu. Exploiting spatial sparsity for event cameras with visual transformers. In *IEEE Int. Conf. Image Process. (ICIP)*, 2022.

[54] Wenming Weng, Yueyi Zhang, and Zhiwei Xiong. Event-based video reconstruction using transformer. In *Int. Conf. Comput. Vis. (ICCV)*, 2021.

[55] Man Yao, Huanhuan Gao, Guangshe Zhao, Dingheng Wang, Yihan Lin, Zhaoxu Yang, and Guoqi Li. Temporal-wise attention spiking neural networks for event streams classification. *Int. Conf. Comput. Vis. (ICCV)*, 2021.

[56] Friedemann Zenke and Surya Ganguli. SuperSpike: Supervised learning in multilayer spiking neural networks. *Neural Computation*, 2018.

[57] Jiqing Zhang, Bo Dong, Haiwei Zhang, Jianchuan Ding, Felix Heide, Baocai Yin, and Xin Yang. Spiking transformers for event-based single object tracking. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2022.

[58] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Unsupervised event-based learning of optical flow, depth, and egomotion. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2019.