Supplementary Material - Recurrent Vision Transformers for Object Detection with Event Cameras

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

1. Relationship to temporal graph neural networks (TGNNs)

The RVT backbone can be understood as an instance of a discrete-time dynamic graph (DTDG) (section 4.6.1 in [4]). A DTDG can be derived from continuous-time dynamic graph (CTDG) by simply discretizing time. One advantage of DTDGs is that they are better suited to contemporary hardware (GPU, TPU, IPU, etc.) than CTDGs.

More specifically, the proposed RVT backbone is an instance of a DTDG with a fixed number of persistent temporal nodes (given a certain input resolution). In our hierarchy, we have four layers of nodes with temporal history (LSTMs in each stage). From here on, there are two different interpretations: (1) The layers up to the LSTMs are a carefully crafted message-passing algorithm interacting directly with the nodes from the previous LSTM nodes closer to the input; or (2) the strided conv layer creates a smaller set of nodes. Block-SA subsequently operates on these nodes by creating a new set of edges (fully graphs connected in local windows) before applying message-passing via self-attention. The same principle applies to Grid-SA which instead creates fully connected graphs in a global dilated grid. The resulting node features are directly used as input to the temporal nodes (LSTM) that also use the states from the previous timestamp to update the node features.

Not every DTDG nor CTDG performs well for event-based vision. In our work, we constrain the design specifically for the problem to get a better trade-off for task performance and inference speed.

2. Data Pipeline

2.1. Mixed Batching Strategy

Two common ways of training recurrent neural networks is using either backpropagation through time (BPTT) or truncated BPTT (TBPTT). This section describes a strategy that combines both.

BPTT Dataloading For BPTT, the dataloader typically samples short sequences without replacement from the dataset. As a consequence, the model cannot use an initialized hidden state from earlier parts of the full sequence. This has the advantage that we can perform more aggressive data augmentation during training. However, the downside is that the model will fail to generalize to longer sequences. One possible solution to this problem is training with TBPTT.

Dataloading Strategy	mAP	AP_{50}	AP_{75}
BPTT	38.8	66.0	39.5
TBPTT	44.1	72.0	46.1
Mixed	**46.0**	**72.3**	**49.4**

Table 1. **Dataloading Strategy**. Combining BPTT and TBPTT (Mixed) yields the best results on the 1 Mpx validation set.

TBPTT Dataloading For TBPTT, the dataloader again extracts short sequences. This time, these short sequences are consecutive sequences from a longer sequences that usually cannot be loaded into RAM. Now we can train the model with initialized, detached hidden states from the previous optimization step. This allows the model to generalize to longer sequences but also precludes some data augmentation techniques. For example, zoom-in augmentation on the whole sequence becomes impractical because we could remove too many labels in the process. Finally, it can also lead to training instabilities and overfitting because the model is optimized on samples from the same sequences for many training steps.

Mixing BPTT and TBPTT What we found to work best is to create two dataloaders: First, a dataloader that randomly samples short sequences from the dataset. This is useful for training with BPTT and improves the diversity of samples in the batch. This dataloader also applies the full set of data augmentations. Second, a dataloader that collates data from iterating through whole sequences. This dataloader is used for training with TBPTT and improves the capability of the model to generalize to sequences longer

Augmentation	Probability	Magnitude	
		min	max
horizontal flip	0.5		
apply zoom	0.8		
zoom-in	0.8	1	1.5
zoom-out	1-P(zoom-in)	1	1.2

Table 2. **Data Augmentation Parameters**. The probability defines the Bernoulli distribution from which we draw the decision whether to apply this augmentation on a given sample.

than the training sequence length. In this case, we do not apply zoom-in augmentation.

Tab. 1 compares the performance of the RVT-B model on the validation set of the 1 Mpx dataset using the different dataloading strategies. The performance of BPTT can be improved by increasing the sequence length at the cost of increased memory consumption and training time. Instead, the mixed dataloading strategy allows training with a short sequence length (here 5) while enjoying stable training with improved detection performance.

In practice, on each GPU, half of the batch is created with the BPTT dataloader and the other half with the TBPTT dataloader. Both batches are separately loaded onto the GPU before being collated and fed to the model. The model then dynamically resets the hidden states based on the information whether each sample in the batch originates from the BPTT or TBPTT dataloader.

2.2. Data Augmentation Details

We apply three data augmentation techniques to train our models from scratch. Table 2 summarizes the probability of each augmentation being used on an individual sample.

For each sample, we may apply horizontal flipping and also apply a zoom augmentation subsequently. For the zoom augmentation we draw from a Bernoulli distribution to indicate whether we apply the augmentation at all. If zoom augmentation shall be applied, we randomly choose between zoom-in or zoom-out augmentation based on the respective probability. For the zoom augmentations, there is also the parameter that defines the *magnitude* with which the augmentation is applied. A magnitude of 1 means that no zoom is applied while a magnitude greater than 1 indicates how strongly zoom-in or zoom-out is applied with respect to the original resolution. The magnitude that we finally apply is drawn from a continuous uniform distribution with bounds *min* and *max*.

3. Additional Training Details

The attention window size for all stages is set to 8×10 for the Gen1 dataset and 6×10 for the 1 Mpx dataset. We also found that it can help to maintain an exponential moving average (EMA) of trained parameters during training. Specifically, for the 1 Mpx dataset, we maintain a separate set of parameters using an EMA smoothing factor of 0.001. This means that for each iteration the EMA parameters are computed from a convex combination of 0.01% of the new weights and 99.99% of the previous weights. Overall, using EMA yields more stable validation results, but similar performance can be achieved without EMA.

LSTM residual	mAP	AP_{50}	AP_{75}
without residual	**47.6**	**70.1**	**52.6**
with residual	46.0	69.8	50.3

Table 3. **LSTM with and without residual connection**. Using a skip connection over the LSTM cells leads to worse results on the Gen1 validation set.

4. Additional Experiments

This section provides two additional experiments that did not fit into the main paper. First, we briefly discuss an ablation on the possibility of using residual LSTM layers. Second, we follow with a qualitative study of cross-dataset generalization using our model trained on the 1 Mpx dataset.

4.1. Residual LSTM Ablation

Our model employs LSTM cells [3] without skip/residual connections in the model. We also experimented with adding skip connections to the LSTM cells on the Gen1 [1] dataset. Table 3 shows that adding a residual connection to the LSTM cells leads to worse results. We hypothesize that this residual connection hampers the LSTM's ability to control the mixture of incoming (current timestep) and retained temporal features (previous timesteps). For example, it would be difficult for the residual-LSTM combination to ignore the incoming feature because the output of the LSTM is simply added to this feature. Without the residual connection, the LSTM could simply set the input gate to 0 to ignore the input.

4.2. Cross-Dataset Generalization: From 1 Mpx to DSEC

DSEC is a dataset that features event cameras and global shutter cameras close to each other. Unlike the 1 Mpx dataset [5] which was recorded mostly urban scenarios in Paris, DSEC [2] provides recordings from urban and rural regions in Switzerland. Furthermore, the event camera used in the DSEC dataset is a Gen 3 prophesee event camera instead of a Gen 4 camera as in the 1 Mpx dataset. In this section, we qualitatively show that our model can be successfully deployed in a different environment and using different event cameras.

We deploy RVT-B, trained on the 1 Mpx dataset , on several sequences of the DSEC dataset to qualitatively assess the cross-dataset generalization. While DSEC does not yet provide object detection labels, we can visually assess the quality of the detections by using the provided calibration files to map the frames of the global shutter camera to the event camera view.

Figure 1 and 2 show predictions of our model together with the images closest in time to the detections. Figure 1 shows our model can successfully detect cars in mountainous environments. In particular, Figure 1 (a) shows an HDR scene where the global shutter frame is overexposed such that the approaching car is barely visible. Due to the high dynamic range of the event camera, our model can detect the approaching car without any difficulty. Figure 2 features more urban environments where our model also manages to detect objects correctly.

Discussion of Failure Cases By and large, our model can successfully detect objects on DSEC even though it has only been trained on the 1 Mpx dataset. Still, we found failure cases that might stem from distribution shift between the datasets. For example, Figure 2 (c) shows the erroneous detection of a two-wheeler that instead is a pillar on the street.

Overall, the model is good at detecting cars but is less confident and accurate at detecting two-wheelers and pedestrians. This effect likely stems from the fact that the 1 Mpx dataset has almost twice as many car labels as pedestrian and two-wheeler labels combined.

5. Dataset Licenses

Gen1 [1] "Prophesee Gen1 Automotive Detection Dataset License Terms and Conditions": https://www.prophesee.ai/2020/01/24/prophesee-gen1-automotive-detection-dataset/

1 Mpx [5] "Prophesee 1MegaPixel Automotive Detection Dataset License Terms and Conditions": https://www.prophesee.ai/2020/11/24/automotive-megapixel-event-based-dataset/

DSEC [2] "Creative Commons Attribution-ShareAlike 4.0 International public license (CC BY-SA 4.0)": https://dsec.ifi.uzh.ch/

References

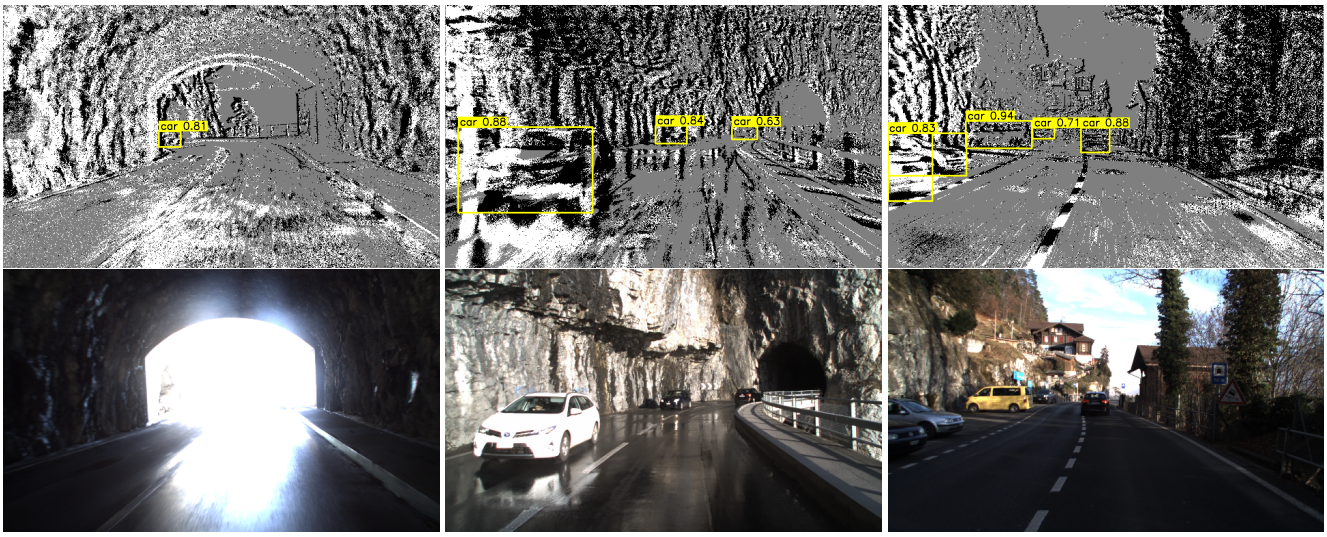
[1] Pierre de Tournemire, Davide Nitti, Etienne Perot, Davide Migliore, and Amos Sironi. A large scale event-based detection dataset for automotive, 2020.

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

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

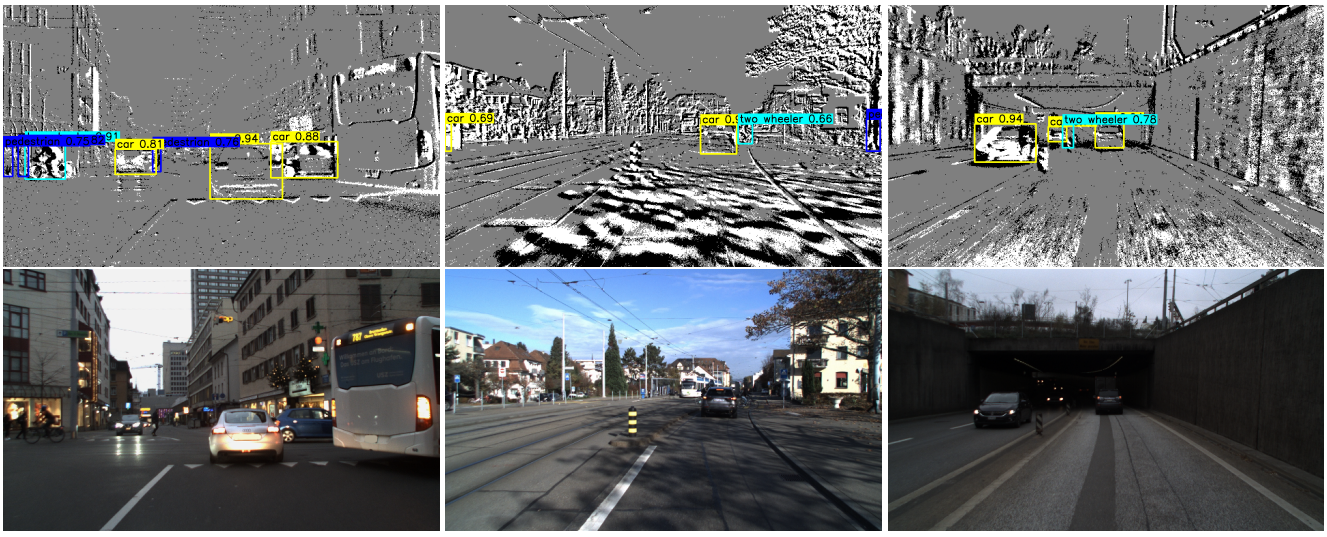
[4] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *J. Mach. Learning Research (JMLR)*, 2020.

[5] 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.



(a) (b) (c)

Figure 1. Prediction examples on the DSEC dataset featuring a **mountainous environment**. Frames are shown only for visualization purposes and are not used by the model. Column (a) shows a typical high-dynamic range (HDR) scenario where the vehicle is exiting a tunnel with a car approaching from outside the tunnel. The HDR capabilities of the event cameras enables our model to detect the approaching car. Column (b) shows a scenario with a wet road and challenging reflections.



(a) (b) (c)

Figure 2. Prediction examples on the DSEC dataset featuring a **(sub-)urban environments**. Frames are shown only for visualization purposes and are not used by the model. Column (a) illustrates a typical urban situation where pedestrians, two-wheelers, cars and other road users occupy the street simultaneously. In column (c), we show a failure case of our model where a street pillar is erroneously detected and classified as a two-wheeler.