Unsupervised Optical Flow Estimation with Dynamic Timing Representation for Spike Camera
Supplementary Material

1 Fine-tune on Real Street Scenes

Most of supervised optical flow estimation methods use simulated ground truth due to the lack of labeled training data in real-world scenes. However, these models may degrade when they are tested on data captured in real-world scenes for the gap in data distribution between simulated data and real data. However, our unsupervised method can overcome the limitation since it does not require the ground truth for training. In more detail, it can fine-tune the model which is pre-trained from SPIFT dataset on the data of real street scenes. As illustrated in Figure 1, the visual quality of optical flow estimated by our fine-tuned model is significantly better than the supervised methods.

1.1 Fine-tune Data

We collect some real data in street scenes. Our unsupervised model which is trained from SPIFT dataset is fine-tuned on one viewing angle of street scenes. Since the spike camera has high temporal resolution, the data for fine-tuning is cut into 4989 clips of $\Delta t = 40$ setting to generate high-speed scenes.

Other viewing angles of street scenes are used to evaluate the fine-tuned model. The test data is also clipped on the $\Delta t = 40$ setting as the same as the data for fine-tuning. In addition to real street scenes, we also collect some other scenes to evaluate the fine-tuned model.

Note that the reason we choose $\Delta t = 40$ is that the motions in the SPIFT dataset are generally larger than the real scenes. If we set Δt too small, the amplitude of motion is not very significant and everything could be much easier.

1.2 Fine-tune Details

We use the best unsupervised model of $\Delta t = 10$ which is trained on the SPIFT dataset as the pre-trained model and fine-tune it for 6 epochs. The learning rate is set to $1e - 4$ and the batch size is 1. The details of our proposed unsupervised loss are the same as the settings during the training on SPIFT dataset.

1.3 Qualitative Results

In Figure 1, we show the qualitative results of our fine-tuned model and other methods on test data. The model is fine-tuned for 6 epochs. Compared with other methods, our method achieves impressive results after fine-tuning.

For SCFlow, the boundary of the predicted motion region is blurry. For PWC(variant), it can not estimate the direction of vehicles. Moreover, the static regions of the optical flow estimated by them are not clean. The USFlow model trained in supervised can not predict significant motions. It is worth noting that our model is trained in unsupervised, so it can be fine-tuned on real data without

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

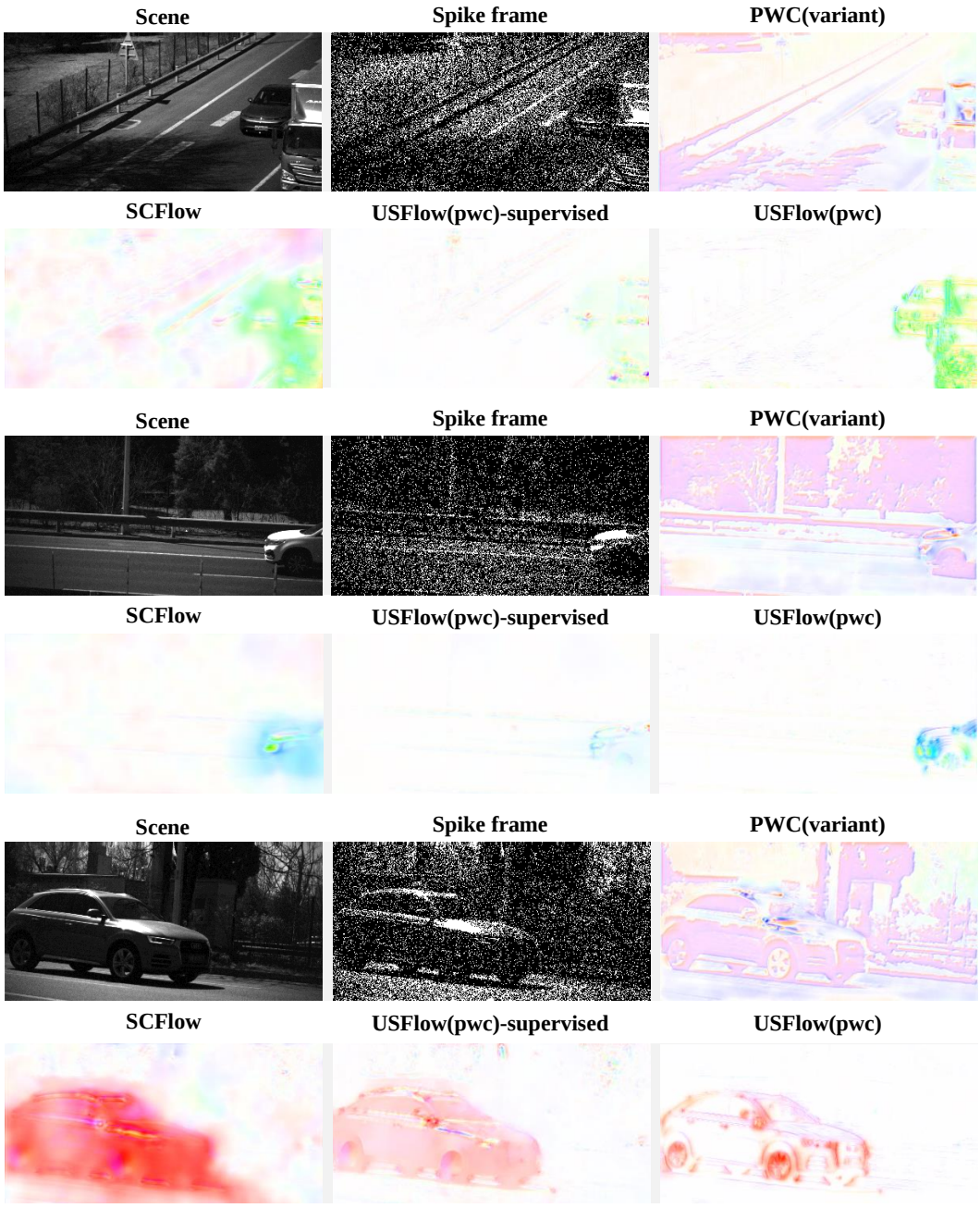

Figure 1: Qualitative results of fine-tune evaluation on real data. Note that 'USFlow(pwc)' is the fine-tuned model. The colormap used in visualization refers to Middlebury [1].

ground truth to eliminate the challenges caused by the domain gap between simulated data and real data.

Furthermore, we provide a video, named *Optical Flow.mp4*, to dynamically display the optical flow of one test perspective.

2 Weights Analysis and Interval Statistics

We design two strategies to estimate the light intensity for high-speed regions and low-speed regions from spike streams. In low-speed regions, we count spikes in a relatively large time span of the spike stream to utilize longer-duration information. In experiments, we set two specific time windows, and

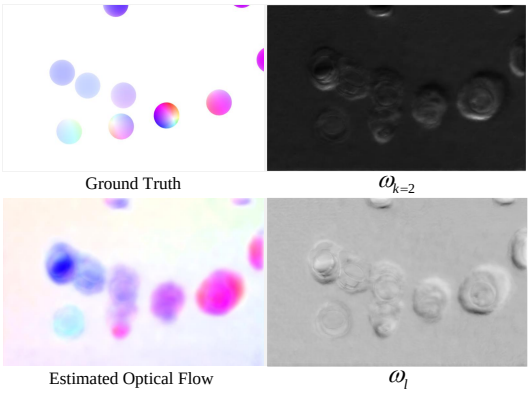

Figure 2: The visualization of the weights in our unsupervised loss.

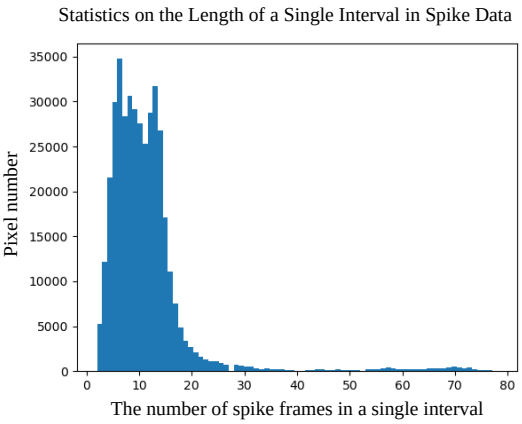

Figure 3: Statistics on the length of a single interval in spike data.

the half lengths of them are $D_s = 40$ and $D_l = 100$ respectively. The 40 and 100 mean the number of spike frames contained in the time windows. In high-speed regions, we estimate light intensity in multi-intervals. Specifically, we set the $k = 1$ and $k = 2$ in experiments. It means that we calculate light intensity in one and three intervals respectively. In Figure 2, we show a case that contains a few balls spinning at high speed. Figure 2 also shows the visualization of weights which play a major role in the shown case. The weights are learned from the estimated optical flow. They indeed have a different emphasis on the regions with different speeds, which is in line with our design purpose. In more detail, the $\omega_{k=2}$ emphasizes the marginal area of the balls spinning at high speed. The ω_l emphasizes the static background and the smooth surface of the balls. Note that the surface of the balls is totally smooth in PHM dataset, so it would be regarded as static region.

As shown in Figure 3, we also count the lengths of spike intervals in spike data. It is obvious that the length of most spike intervals is concentrated in the range of 0 to 20. When k is set to 1 and 2, we use 1 and 3 intervals to calculate light intensity, respectively. Therefore, the corresponding lengths of them are much shorter than the lengths of time windows used in low-speed regions.

3 Case Study on Input Representation

For the output of each dilated convolution layer, we average across the feature channel and scale each feature map to [0,255] for visualization. Figure 4 presents the visualization of the feature map of each layer. Note that the model is unsupervised trained under $\Delta t = 10$ setting.

As we mentioned before, the higher the level is, the larger receptive field the dilated convolution has. As illustrated in Figure 4, the feature map of level 1 is very noisy since the receptive field is small and it is like summarizing the spike stream with a small time window. A small time window cannot provide enough information. As the level increases, we can find out the feature map becomes more dilated and sharp, *i.e.* level 3. However, as the level continues increasing, the visual quality of feature maps degrades. The contrast level 6 is lower than level 5. This is because the large receptive field might smooth the light intensity. It can be regarded as like summarizing the spike stream with a large time window, which may include some redundant information.

4 SSES Dataset

Collecting real data sets in the real environment is costly, and it is impossible to simulate high-cost, complex, and diverse application scenarios. For example, vehicle collisions, pedestrian-vehicle accident scenarios under complex traffic situations. The cost of trial and error is immeasurable, and moral law does not allow it. Considering the above reasons, a synthesis spike-based dataset for the autonomous driving environment is constructed based on CARLAR [2] for different complex scenarios. It mainly includes scenarios such as various collision scenarios, driving under bad weather,

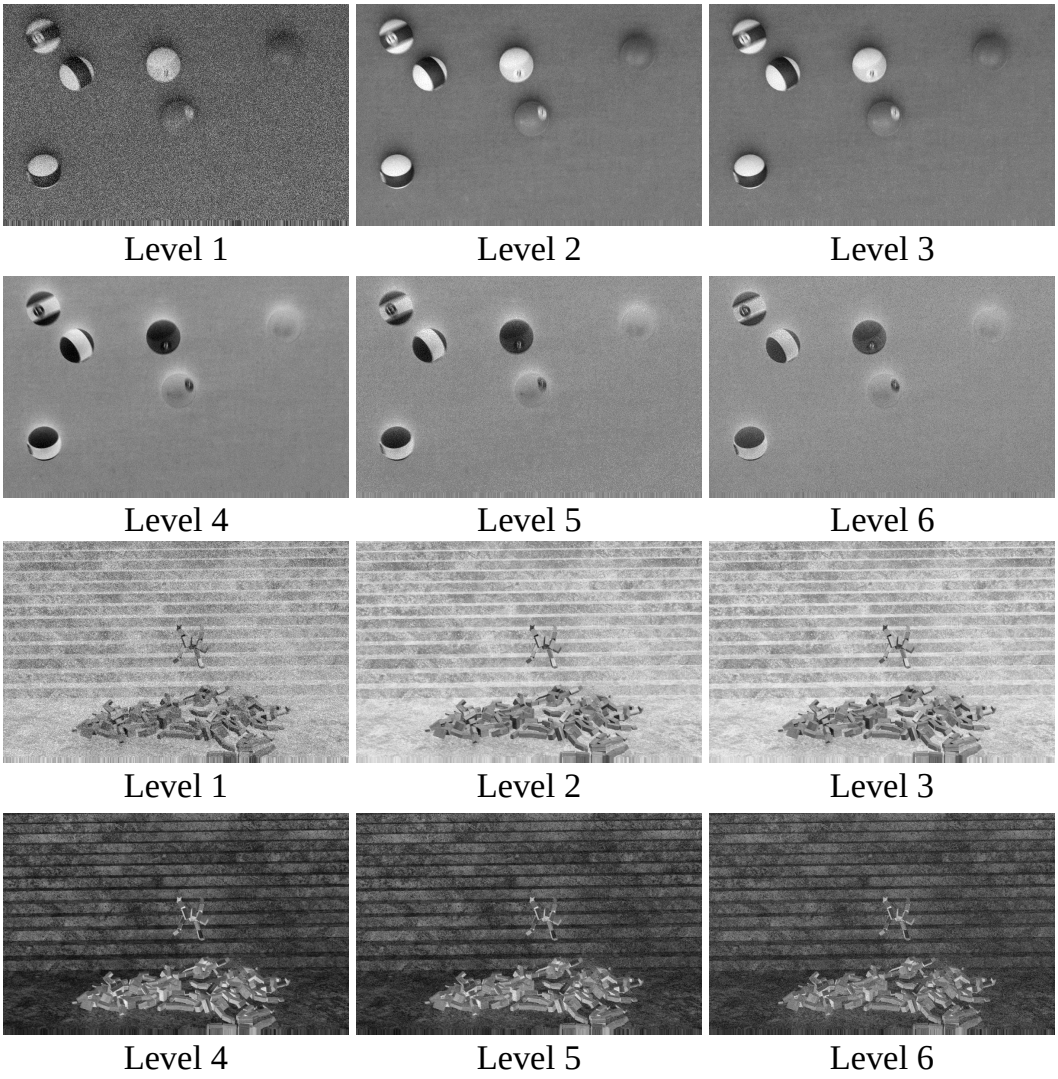

Figure 4: Visualization output of feature map of each layer.

and driving scenarios where road markings are difficult to identify. As shown in Figure 5, we show some sample cases of SSES dataset.

Detailed descriptions of ten scenarios in SSES dataset are listed below. Note that the scenarios are recorded by the Front-Facing Camera of the target car. Other information can be found in Tabel 1.

Pedestrian crossing the road: target car is driving at $15m/s$ on straights, and the pedestrian suddenly runs out from behind obstacles on the side of the road and cross the sidewalk at $6m/s$, and the two collide without braking.

Multiple pedestrians crossing the road: target car is driving at $10m/s$ on straights, and two pedestrians suddenly cross the sidewalk at $6m/s$, and the three collide without braking. Note the vision of the target car is partially blocked by a passing car.

Cars in T-junction: Because the other car do not obey the traffic lights, the two cars drove from different directions to the T-junction and collided at the junction without braking. The target car drove at $16m/s$ and the other car drove at $10m/s$.

Bicycle in T-junction: When the car is turning at a T-junction at $12m/s$, the bicycle crosses the road at $5m/s$ from the blind spot of vision, and the two collide without braking.

Figure 5: The snapshots of ten different extreme scenarios in SSES dataset.

Table 1: Settings of ten scenarios in SSES dataset. The maps are provided by the CARLA itself.

index	scene	weather	map	duration (s)
1	pedestrian crossing the road	rain	Town 1	0.50
2	two pedestrian crossing the road	sunny	Town 2	1.00
3	cars in T-junction	rain	Town 1	1.20
4	cars in T-junction	sunny	Town 1	1.20
5	bicycle in T-junction	rain	Town 1	0.75
6	bicycle in T-junction	sunny	Town 1	0.75
7	cars in crossroad	sunny	Town 4	0.80
8	cars in crossroad	sunny	Town 4	0.80
9	pedestrian in crossroad	sunny	Town 5	0.66
10	pedestrian crossing the highway	sunny	Town 4	0.50

Cars in crossroad: The target car goes straight at $13m/s$, however, there is another car turning toward the target car at $10m/s$ because it does not obey the traffic lights. Two cars collide at the crossroad without braking.

Pedestrian in crossroad: When the target car turns at the crossroad at $15m/s$, pedestrians cross the road at $6.2m/s$ from a blind spot of view, and the two collide without braking.

Pedestrian crossing the highway: The target car is driving on the highway at $33m/s$, and the pedestrian runs out from on the side of the road and cross the sidewalk at $5m/s$, and the two collide without braking.

The ground truth of optical flow and RGB frames are 500 fps and spike frames are 40K fps. We first increase the frame rate of RGB frames to 40K fps through a flow-based interpolation method in OpenCV and then generate spikes by treating pixel value $(0 \sim 255)$ as incoming light intensity and simulating the integrate-and-fire mechanism mentioned in Section 3.1 of paper. The θ is set to 400.

Note that we will release the SSES dataset after the publication of the paper.

5 Training Details

All training images are cropped to 500×800 size. Adam optimizer is used for the training process. In USFlow(pwc), the initial learning rate is scaled by γ following *milestones*. While in USFlow(raft), `torch.optim.lr_scheduler.OneCycleLR` is used with default settings in [5].

More detailed hyper-parameter in training process can be found in Tabel 2.

Table 2: Hyper-parameters in training process.

	Hyper-parameter	Supervised		Unsupervised	
		$\Delta t = 10$	$\Delta t = 20$	$\Delta t = 10$	$\Delta t = 20$
USFlow (pwc)	learning rate	0.0004	0.0006	0.0004	
	batch size	4			
	training epochs	60		25	
	γ (Adam optimizer)	0.8			
	$milestones$ (Adam optimizer)	[5,10,20,30,40,50,70]		[10,30,40,50]	
	N(half window length)	12 (following settings in [3])			
	k	-	-	1,2	
	D_s	-	-	40	
	D_l	-	-	100	
	λ	-	-	50000	
USFlow (raft)	learning rate	0.005		-	-
	batch size	4		-	-
	training epochs	30		-	-
	N(half window length)	12 (following settings in [3])		-	-

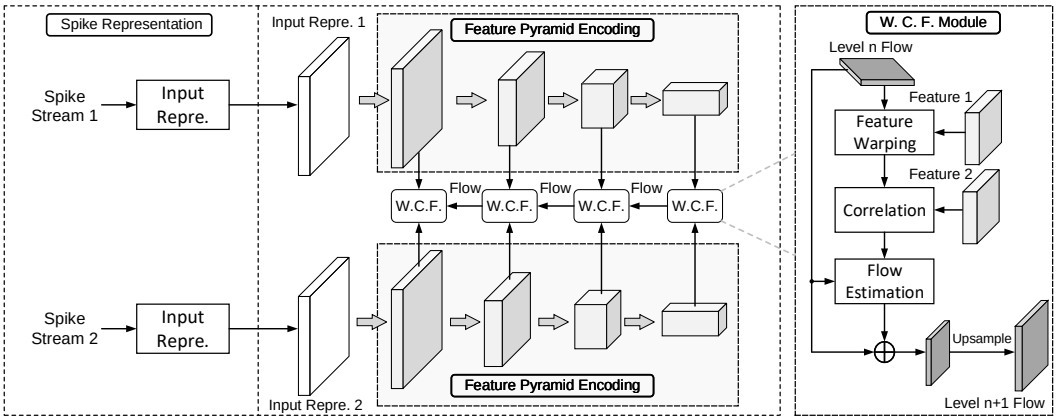

Figure 6: The architecture of PWC(variant).

6 Architecture

6.1 Backbones

The backbone, PWC(variant)[3], is illustrated in Figure 6. We build a 4-level feature pyramid $\{E_i^l(\mathbf{x})\}_{l=1}^4, i = 0, 1$, which denotes the feature for describing the scene at time t_i at l-th level. The feature pyramid has 32, 64, 96, and 128 channels at each level respectively.

We estimate the optical flow from the higher level to the lower level in the pyramid. We refer to the well-known PWC-Net[4] to design the decoder. At l-th level, we firstly warp $E_1^l(\mathbf{s})$ via the current estimated flow $\mathbf{f}_{t_0,t_1}^{l+1}(\mathbf{S})$:

$$E_{1,\mathbf{f}_{t_0,t_1}}^l(\mathbf{x}) = E_1^l\big(\mathbf{x} + \mathbf{f}_{t_0,t_1}^{l+1}(\mathbf{x})\big), \tag{1}$$

where we use bilinear interpolation for the warping operation. We use the features to build a correlation volume[4] to describe the similarity between the features. The correlation volume represents the potential displacements between the two frames, and we normalize the feature in each channel, which can be formulated as:

$$\mathbf{C}^l(\mathbf{x}, \mathbf{m}) = \left\langle \frac{E_0^l(\mathbf{x}) - \mu_0^l}{\sigma_0^l}, \frac{E_{1,\mathrm{w}}^l(\mathbf{x}+\mathbf{m}) - \mu_{1,\mathrm{w}}^l}{\sigma_{1,\mathrm{w}}^l} \right\rangle, \tag{2}$$

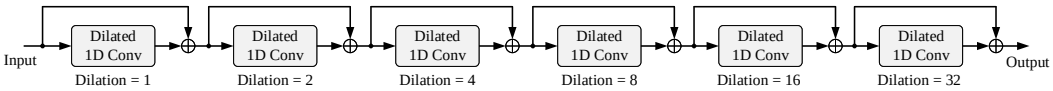

Figure 7: The architecture of dilated convolutions.

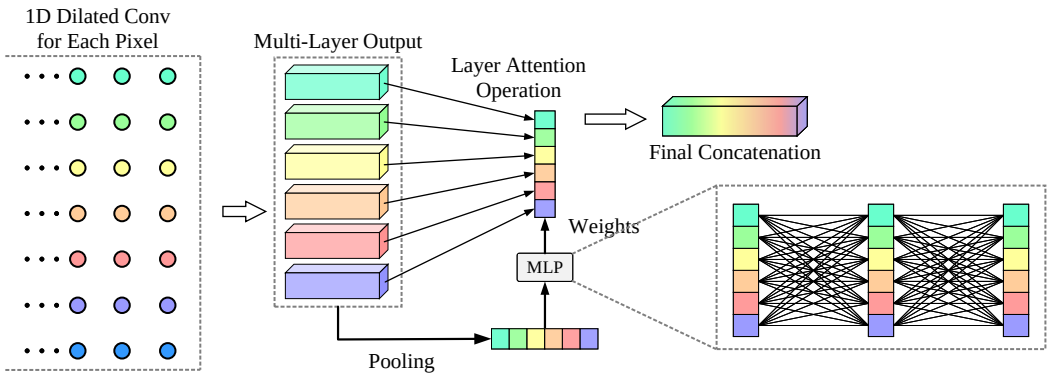

Figure 8: The architecture of layer attention.

where $\mathbf{m}$ represents the displacement between the two features, $\mathbf{f}_{t_0,t_1}$ is written as $\mathbf{f}$ for simplicity and $\langle\cdot\rangle$ is the channel-wise inner product operation. μ and σ is the mean and standard deviation value corresponding to the feature with the same subscript.

The correlation and the feature extracted from the former spike stream at the current level are input to the weight-shared flow estimator. A 1×1 convolution is employed to adjust the channel numbers at different levels to be 32. The flow estimator consisting of cascaded convolutional layers predicts the residual flow. The refined flow is then upsampled via bilinear kernel as the final output of current level.

For RAFT, it has three main components. A feature encoder extracts features from both input spike streams, along with a context encoder that extracts features only from the first input spike stream. A correlation layer that constructs a 4D correlation volume by taking the inner product of all pairs of feature vectors. An update operator based on Conv-GRU recurrently updates optical flow by using the current estimation to look up values from the set of correlation volumes. Note that we adopt an official small version of RAFT, of which the feature dimension is largely reduced. Please move to paper[5] and get more detailed descriptions. For implementation, we follow the official code, whose link is https://github.com/princeton-vl/RAFT.

6.2 Input Representation

To avoid gradient explosion or vanishing, we adopt the residual 1D-dilated convolution block as shown in Figure 7, which includes a shortcut connection that adds the input and output of convolution together. The output feature dimension for each block is 1. We have 6 1D-dilated convolution blocks.

The architecture of the layer attention module is presented in Figure 8. Since we have 6 outputs after dilated convolution, the feature dimension of the hidden layer of MLP is also set to 6.

7 Evaluation Metric

Average End-point Error (AEE is used as an evaluation metric for optical flow estimation. It measures the mean distance between the predicted flow $\mathbf{f}_{pred}$ and the ground truth $\mathbf{f}_{gt}$ provided by the dataset. The AEE is computed as follows:

$$\mathrm{AEE} = \frac{1}{HW} \sum_{H} \sum_{W} \|\mathbf{f}_{\mathrm{pred}} - \mathbf{f}_{\mathrm{gt}}\|_2 \tag{3}$$

References

[1] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision (IJCV)*, 92(1):1–31, 2011.

[2] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Annual Conference on Robot Learning (CoRL)*, pages 1–16, 2017.

[3] Liwen Hu, Rui Zhao, Ziluo Ding, Ruiqin Xiong, Lei Ma, and Tiejun Huang. SCFlow: Optical flow estimation for spiking camera. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[4] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8934–8943, 2018.

[5] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision (ECCV)*, pages 402–419, 2020.