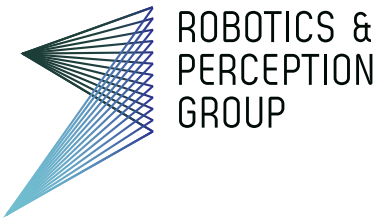
University of Zurich[UZH]
Department of Informatics

University of Zurich[UZH]
ETH zürich
Institute of Neuroinformatics



Ivan Alberico

Learning to Generate Events using Spiking Neural Networks

Semester Thesis

Robotics and Perception Group
University of Zurich

Supervision

Daniel Gehrig
Mathias Gehrig
Prof. Dr. Davide Scaramuzza

Jan 2022

Contents

Abstract

The recent advent of event cameras in computer vision applications has significantly increased the performance of traditional methods, by leveraging the outstanding advantages of these novel sensors over conventional cameras. If on one hand the event-based techniques have achieved impressive results, on the other the same methods are nonetheless limited by the scarce amount of event data required for training. In this work, we propose a learning-based solution relying on Spiking Neural Networks that addresses this issue by generating event data starting from the huge amount of pre-existing video datasets recorded with conventional cameras. The Spiking Neural Networks leverage the asynchronous nature of events to generate event data in an end-to-end fashion, starting from high temporal resolution videos. We evaluate the method on different levels: (i) a visual inspection of the generated events compared to the ground truth ones; (ii) an in-depth analysis of the event rates of the generated events; (iii) an object classification task validated on the N-Caltech101 dataset.

Nomenclature

Notation

$\mathbf{u}(t)$	neuron's state/membrane potential
$\mathbf{s}_i(t)$	i^{th} input spike train to a neuron
$\mathbf{s}(t)$	neuron's output spike train
$\hat{\mathbf{s}}(t)$	target spike train
$t_i^{(f)}$	time of the f^{th} spike of the i^{th} input
$\epsilon(\cdot)$	spike response kernel
$\nu(\cdot)$	refractory response kernel
w_i	synaptic weight of the i^{th} input spike train
$\mathbf{a}_i(t)$	spike response signal

Acronyms and Abbreviations

HDR	High Dynamic Range
ANN	Artificial Neural Network
SNN	Spiking Neural Network
SRM	Spike Response Model
PSP	Post Synaptic Potential

Chapter 1

Introduction

Visual perception represents a core aspect of modern engineering, especially in areas like robotics or artificial intelligence. Standard cameras historically represent the main sensors involved in the vast majority of applications and although their remarkable achievements, they still present limitations that make them unfeasible to operate under certain circumstances. Lately, attention has shifted onto a novel type of sensor which is achieving resounding success in computer vision: event cameras.

Event cameras are innovative sensors that offer significant advantages with respect to standard cameras. Contrary to traditional cameras, which operate by synchronously opening and closing a shutter to let light in and capture images at fixed rates, event cameras are build in a totally different way. In event cameras each pixel is a light-sensitive sensor that operates independently and that is programmed to collect data whenever the amount of light it receives either increases or decreases by a certain quantity. For this reason, event cameras are more responsive than conventional cameras, leading to a significantly higher temporal resolution and a low latency, in the order of microseconds. In addition to that, they are characterized by a high dynamic range (HDR) and no motion blur, allowing them to be suitable in more challenging applications that involve rapid motion and low light.

Research in this area has shown that computer vision techniques based on event data have achieved outstanding results, leading to a growing interest in this field. However, one the main limitations of these methods is that they require a large amount of event data for training, which is not available for mainly two reasons: first, because these sensors have been recently introduced in the market and secondly because they are rather expensive, making them not accessible to everyone. The goal of this project is to design a learning-based solution that addresses this issue by converting any existing frame-based dataset recorded with conventional cameras into synthetic event data. In this way, we can leverage the enormous amount of video datasets to automatically generate events.

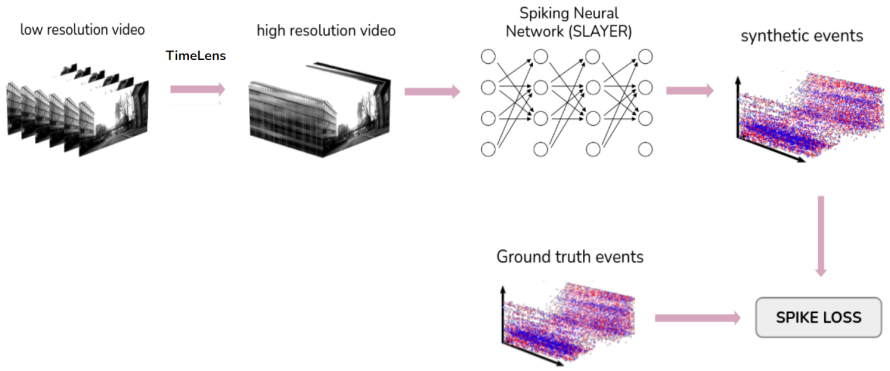


Figure 1.1: General pipeline of the proposed method.

The asynchronous nature of events suggests the use of particular types of networks known as Spiking Neural Networks (SNN), bio-inspired sensors that process information conveyed as temporal spikes rather than continuous numeric values. The way events are generated whenever a a significant brightness change is witnessed, recalls the dynamics of spiking neurons that produce spikes only when their state (membrane potential) reaches a predefined threshold.

The general pipeline of the project can be summarized in the following steps: starting from low resolution videos, Time Lens [13] frame interpolation technique is first deployed to generate high resolution videos which are directly fed into the event generation network. The training process of the Spiking Neural Network is supervised by a spike loss relying on ground truth events. To summarize, the main contributions are:

- We propose a learning-based solution to generate synthetic events starting from video sequences.
- We show that the events generated with this method accurately reproduce the corresponding real events in most of the cases, although being sensitive to noise.
- We evaluate our method on an object classification task and show that models trained on the synthetic events generated with the proposed network perform better than Vid2E [4] in terms of accuracy.

1.1 Related Work

Prior work addressing the issue of scarce event data is already present in literature. The work done in [3] proposes a method to convert existing video dataset to synthetic event data that relies on ESIM [11], an event camera simulator which generates events using an adaptive sampling scheme. The adaptive sampling scheme introduced by [11] is based on the maximum displacement between frames and guarantees high accuracy when modeling fast motion and lower computation in case of slow motion sequences.

The way events are generated in these methods follows the *Event Generation Model*, which can be summarized by the following

$$\Delta \log I\left(\mathbf{u}, t_k\right) = \log I\left(\mathbf{u}, t_k\right) - \log I\left(\mathbf{u}, \Delta t_k\right) \geq p_k C, \tag{1.1}$$

where $I\left(\mathbf{u}, t_k\right)$ represents the brightness at time t_k of pixel $u = (x_k, y_k)^T$. The work done in this project conceptually builds on top of [3], in fact the main steps to achieve the goal are similar. What is different though, is the way events are generated. This method substitutes the *Event Generation Model* with a Spiking Neural Network, which is supervised on real events. The aim of the project is to improve both accuracy and generalization capabilities of the previous methods. The main limitation of these methods is represented by their dependency on predefined values of the contrast threshold C, which does not allow them to generalize on different types of datasets. Training an SNN would, in theory, overcome this issue, in a way that events are not generated in a fixed way, but they rather depend on the specific input we are providing to the network.

Using SNNs to process event data is an approach which is becoming increasingly adopted in research. Several works have proposed to use SNNs to process asynchronous event streams for computer vision application. An example is presented in [9], where the author integrates SNNs and ANNs for efficiently estimating optical flow from sparse event camera outputs. Recently, [5] has investigated the use SNN to predict 3-DOF angular velocity of a rotating event-camera, performing a temporal regression problem starting from event. However, the way we seek to integrate events with SNNs in this work is slightly different from previous methods: the events are not directly fed to the network but they are used to supervise the training procedure via customized loss functions.

Chapter 2

Method

In this section we describe the main building blocks of the proposed method for converting video to synthetic events. The process can be divided in two main blocks: a data pre-processing stage followed by the training of a tailored network for event generation. For the first part, we leverage a recent event-base frame interpolation technique [13] to convert low frame rate to high frame rate video. The upsampled datasets are then used to train a Spiking Neural Network that learns how to generate events in an end-to-end fashion.

Before diving into the details of how events are generated, in Section 2.1 the dynamical model of a spiking neuron is described. Understanding how information is propagated across a Spiking Neural Network is of paramount importance to understand how events are generated within the network.

2.1 Dynamics of a Spiking Neuron

In this section we present the dynamical model of a spiking neuron and how it is integrated to our model. Spiking neurons differ from traditional neurons mainly for the fact that they are not activated at each cycle, but they transmit information only when their state is above a certain threshold. In general, these neurons take spikes as both input and output, contributing to the neuron's internal state over time. In this work, we will follow the model defined by [12], which uses a simple spiking neuron model known as the Spike Response Model (SRM).

Given an input train of spikes defined as $s_i(t) = \sum_f \delta(t - t_i^{(f)})$, we first generate a *spike response signal* $a_i(t)$ by convolving $s_i(t)$ with the spike response kernel $\epsilon(\cdot)$. In a similar way, the refractory response of a neuron is modeled as $(\nu * s)(t)$, where $\nu(\cdot)$ is the refractory kernel and $s(t)$ is the neuron's output spike train. The spike response and the refractory response can be thought respectively as a feed-forward and a feed-back term acting on each spiking neuron of the network.

Each spike response signal is then scaled by a synaptic weight w_i, representing the learnable parameters of the SNN, generating a Post Synaptic Potential (PSP). The contribution of each spike to the PSP, namely its magnitude and

sign, is fully determined by the synaptic weight corresponding to the spike. The neuron's state, also referred to as membrane potential $u(t)$, is simply the sum of all post-synaptic potentials and refractory responses:

$$u\,(t) = \sum w_i(\epsilon * s_i)(t) + (\nu * s)(t) = \mathbf{w}^T\mathbf{a}(t) + (\nu * s)(t), \tag{2.1}$$

Output spikes are generated from the membrane potential, whenever the value of $u(t)$ overcomes a predefined threshold θ, defined as membrane threshold. More formally, the spike function $f_s(\cdot)$ is defined as

$$f_s(u) : u \to s, \qquad s(t) := s(t) + \delta(t - t^{f+1}), \tag{2.2}$$

where $t^{f+1} = \min\{t : u(t) = \theta, t > t^f\}$. In the moment immediately after a spike is generated, the neuron tries to bring its membrane potential to the resting potential value, so that the spiking activity is regulated. This self-suppression mechanism is called refractory response. A schematic representation of this whole mechanism is shown in Fig. 2.1.

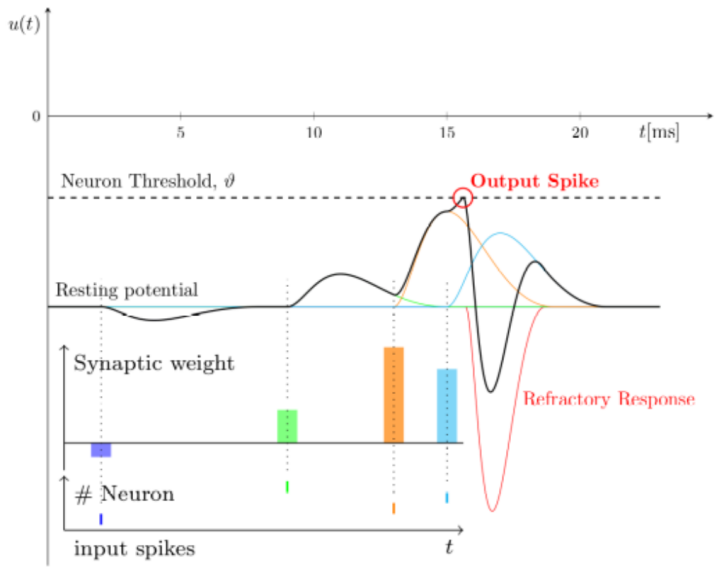


Figure 2.1: Schematic representation of the dynamics of a spiking neuron.

The way input video frames are integrated in this model is rather simple: instead of feeding each video sequence to the spike response kernel, as we would normally do with asynchronous spikes, we assume that the input frames constitute the membrane potential of the very first layer of the network. This assumption holds because the video datasets have been upsampled to a high resolution, so they represent a valid approximation of a continuous signal. In this way, the spike function acts directly on the input frames and events are generated. Nevertheless, the synaptic weights still play a fundamental role, since

they represent the parameters regulating the learning process and they scale the frames' intensities before being thresholded.

2.2 Data pre-processing

The supervised learning nature of the proposed method requires pre-processing on two different levels: on one hand we need to take care of upsampling the input video datasets fed to the network during training, and on the other we need the ground truth events deployed for supervision to be correctly represented.

In general, training neural networks requires a huge amount of data to guarantee that meaningful patterns are learned during the process. In addition to this, the data provided should contain as many different scenes as possible, so that we avoid overfitting to specific types of sequences. In this work, we propose two different datasets to train the network, the HS-ERGB dataset [13] and the Vimeo90k-denoising dataset [14].

2.2.1 Dataset upsampling

The upsampling step of the input data frames represents a core aspect of the event generation process, as it may affect the final quality of the generated events. Differently from [3], in which frames were upsampled using Super SloMo video interpolation method [7], the method proposed in this work relies on Time Lens [13], a novel event-based frame interpolation method that brings together the advantages of synthesis-based and flow-based approaches. While Super SloMo leverages bi-directional optical flow between adjacent frames to warp them at specific timestamps to generate the intermediate images, Time Lens estimates motion from events, rather than frames and thus has several advantages: it is more robust to motion blur and can estimate non-linear motion between frames. Time Lens is deployed to upsample the input video sequences up to 1000 FPS. In this way, we assume that consecutive frames in every sequence have $1ms$ difference in time, which simplifies the whole set-up.

High Speed Events-RGB (HS-ERGB) dataset combines together high-resolution events with RGB images. The hybrid camera setup to record this dataset features a Prophesee Gen4 event camera and a FLIR BlackFly S global shutter RGB camera. The dataset is recorded in a variety of conditions, both indoors and outdoors. The exposure time used to record the sequences is as low as $100\mu s$ for outdoor scenes and up to $1000\mu s$ for indoor scenes. The dataset features frame rates of 160 FPS and includes highly dynamic close scenes with nonlinear motions and far-away scenes featuring mainly camera ego-motion.

Vimeo-90k is a large-scale, high-quality video dataset for low-level video processing. The Vimeo-90k-denoising dataset that we use to train the network is a sub-section of the *septuplet dataset*, consisting of 91.701 7-frame sequences with fixed resolution 448×256, extracted from 39K selected video clips from Vimeo-90K. This dataset is designed to video denoising, deblocking, and super-resolution. To address video denoising, two types of noises are considered: a Gaussian noise with a standard deviation of 0.1, and mixed noises including a 10% salt-and pepper noise in addition to the Gaussian noise. Since this dataset

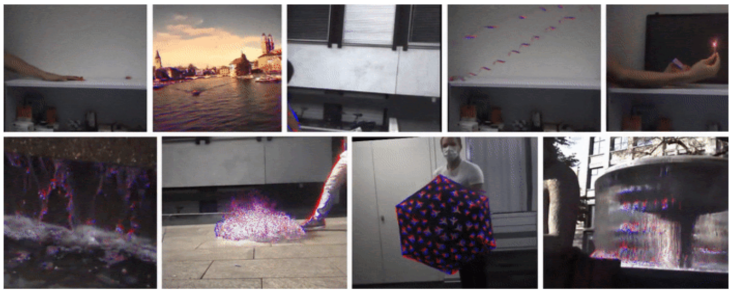


Figure 2.2: Sample sequences from the HS-ERGB dataset.

contains only video frames, the ground truth events used for supervision while training on this dataset are synthetic events generated with Vid2E [3].

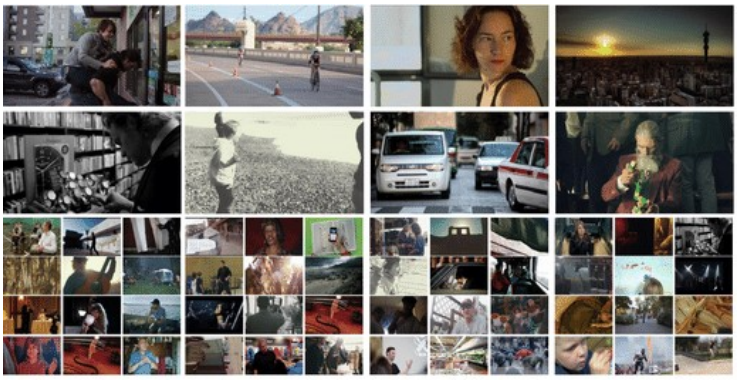


Figure 2.3: Sample sequences from the Vimeo-90k dataset.

2.2.2 Event representation

The way events are represented is a fundamental step when dealing with event-based computer vision methods. An event is formally described as a tuple (x, y, t, p), where x and y are the location of the pixel from which the event was triggered at time t. The polarity p is a binary variable that indicates whether the change in brightness is either positive or negative. In general, events are aggregated in grid-based representations, allowing them to be processed by traditional neural network models.

The work done in [4] specifically addresses the issue of converting asynchronous event-based data into grid-based representations. Their contribution consists in expressing the conversion process through kernel convolutions, quantizations, and projections, where each operation is differentiable.

In this work, events are expressed as tensors of size $[2, W, H, T]$, where W and H refer to the width and height of the corresponding video frames from which they are generated and T being the temporal dimension of the sequence, i.e. the number of frames. In this way, for a given timestamp value, each pixel location of the event tensor is associated to two values accounting for the negative and

positive polarity of the corresponding event. To obtain such representation, a discretization step is performed in which every stream of events is distributed into T equally-sized bins along the temporal dimension.

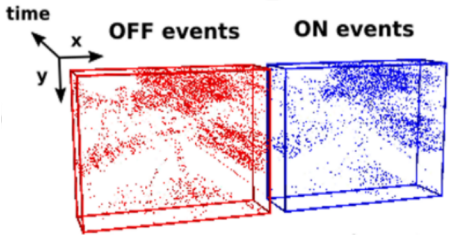


Figure 2.4: Event Spike Tensor representation of a stream of events.

2.3 Network architecture

In this section we present Spike-ESIM, a novel Spiking Neural Network for event generation from video frames. The proposed network is different from traditional Spiking Neural Networks in the way that it does not take as input trains of asynchronous spikes, but rather sequences of upsampled video frames which are fed directly at the level of the membrane potential of the first layer. In this way, we skip the Post Synaptic Potential layer at the initial stage of the network, which normally convolves the input spikes of each node with the temporal $\epsilon(\cdot)$ kernel to generate the potentials.

The network is a branched architecture, with each branch modeling a specific polarity of the output event tensor. Modeling the event generation process with a single network would not be feasible, since we need to account for both positive and negative changes of brightness along the frames. Another important aspect is that the two branches should not have shared parameters during the training. A Siamese-like architecture, for example, where the weights of the branches are shared and trained concurrently, would not be accurate because positive and negative events need to be modeled separately, as they are subject to different contrast thresholds C in the real-world case.

The way the input video frames are fed into the network is the following: starting from a sequence of T consecutive frames, we compute both the positive and negative difference of the intensities along adjacent frames, namely $\Delta I(t)$ and $-\Delta I(t)$, and we feed them into the two separate branches. Each branch is composed of a 3D convolution layer followed by a spike function module, this latter being the one that actually generates events by thresholding the scaled differences of brightness according to the membrane potential hyper-parameter. Each convolution block is a 3D convolution with 3×3 kernel size, padding of size 1, 3 input channels corresponding to the RGB channels of the input video frames and 1 output channel corresponding to the generated events of the specific polarity. The final output of the network is computed by concatenating along the channel dimension the two feature maps formed at the end of each branch.

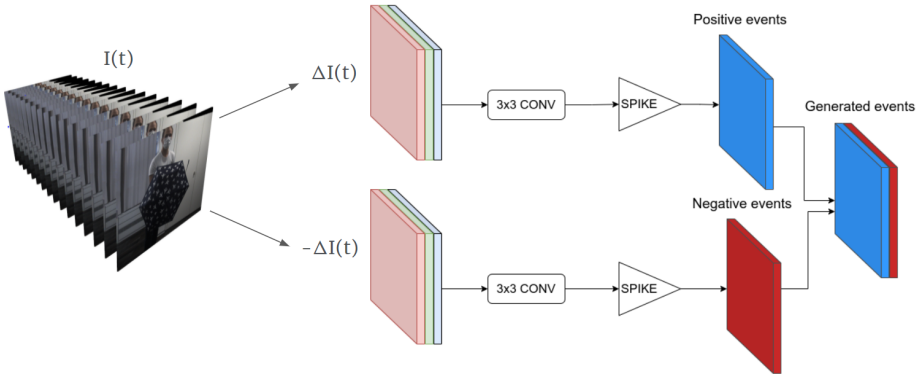


Figure 2.5: Schematic representation of the Spike-ESIM architecture.

2.4 Training Procedure

In this section we discuss the main blocks needed to setup a training with the proposed network and all the details related to it. The training of our SNN is done with the publicly available [1] PyTorch implementation of SLAYER [12].

2.4.1 Backpropagation in Spiking Neural Networks

The main drawback of training Spiking Neural Networks is that the derivative of the spike function is not defined, being this function non-differentiable, which is a major concern when it comes to backpropagate the error from output to input during training. Prior work addressing this issue can be split in two main approaches: a first approach overcomes this problem by introducing an ANN that trains an equivalent shadow network, while a second approach is more oriented to finding functions that approximate the derivative of the spike function.

The SLAYER model [12] addresses this problem by defining the derivative of the spike function as a Probability Density Function (PDF) modeling the change of state of a spiking neuron. This PDF is represented by an exponentially decaying function of the random variable $u(t) - \theta$ in the form

$$\rho(t) = \frac{1}{\alpha} \exp\left(-\beta|u(t) - \theta|\right). \tag{2.3}$$

This probability function $\rho(t) = \rho(u(t) - \theta)$ assumes high values when $u(t)$ is close to θ and it decreases the further the membrane potential $u(t)$ is far away from the membrane threshold.

2.4.2 Loss function

The loss function L used during the training is defined as the time-integral over a period T, this latter representing the temporal dimension of each sequence fed

[1] `https://github.com/bamsumit/slayerPytorch`

into the network, in the form

$$L = \int_0^T l(s^{(n_l)}(t), \hat{s}(t))dt = \frac{1}{2}\int_0^T \left(e^{(n_l)}(s^{(n_l)}(t), \hat{s}(t))\right)^2 dt, \tag{2.4}$$

where $\hat{s}(t)$ is the ground truth event tensor, $s^{(n_l)}(t)$ is the output of the network, $l(s^{(n_l)}(t), \hat{s}(t))$ is the loss at time t and $e^{(n_l)}(s^{(n_l)}(t), \hat{s}(t))$ is the error signal at the final layer n_l. As suggested in [12], to learn a target spike train $\hat{s}(t)$, in this case the ground truth events, is convenient to choose the error signal in the form

$$e^{(n_l)}(t) = \epsilon * \left(s^{(n_l)}(t) - \hat{s}(t)\right) = a^{(n_l)}(t) - \hat{a}(t), \tag{2.5}$$

which computes the loss as an element-wise difference between the generated event tensor and the ground truth events, convoluted with the response kernel.

2.4.3 Experimental setup

The CUDA accelerated deep learning framework implemented by [12] is used to perform the training of the SNN. In our experiments, we use spike response kernels in the form $\epsilon(t) = t/\tau_s \exp(1 - t/\tau_s)\,\Theta(t)$ and a refractory response kernel in the form $\nu(t) = -2\theta \exp(1 - t/\tau_r)\,\Theta(t)$, with $\tau_s = 10$ and $\tau_r = 1$.

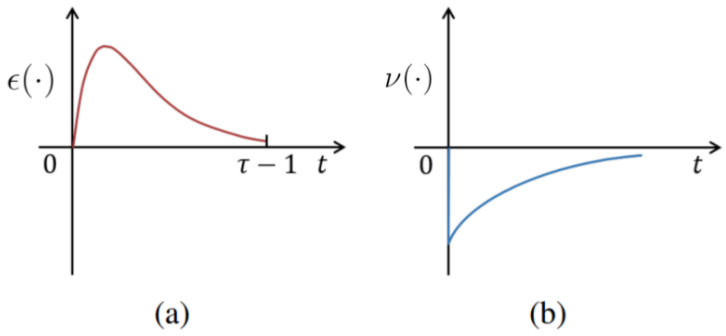


Figure 2.6: Graphical representation of the proposed: (a) spike response kernel (b) refractory kernel.

The optimizer chosen for the training is ADAM [8] with a learning rate of 0.01. Extremely important is the choice of the membrane threshold θ of the SNN, since it significantly affects the performance of the results. A too high threshold results in overlooking some events in the scene, while a too small threshold takes in a lot of details which cannot be properly classified as events. Different values of the membrane threshold were tested: 0.2, 0.7, 1, 2, 3, 5. The best observed value for the Vimeo-90k-denoising is 0.7, while for the HS-ERGB dataset is 1.5.

Training on the Vimeo-90k-denoising dataset is performed considering the full resolution (448×256) of the upsampled input sequences. Every sequence of the

upsampled Vimeo-90k-dataset contains 205 frames. As far as the HS-ERGB dataset is concerned, central 300×300 cropping is performed to the input sequences to reduce the computational burden and to conform to a unified spatial dimension during the training procedure. One of issues encountered during the training on the HS-ERGB dataset concerns the fact that some sequences are extremely long after the upsampling step and therefore unfeasible to be processed all at once. To address this problem, a recurrent implementation of the SNN modules is proposed, which allows to sub-divide every sequence into chunks of 1000 frames each, and to process every chunk sequentially. While processing the different chunks of a sequence, we keep track of the results of the previous stages. In this way we overcome the computational burden of processing extremely long sequences all at once, which is nonetheless unfeasible due to memory and GPU limitations.

Chapter 3

Experiments

In this chapter we provide experimental results to validate the effectiveness of the proposed method. First, a more qualitative evaluation is performed by visually comparing the generated events to the ground truth events. Then, we move on to analyze the event rates of specific sequences to monitor how the number of spikes changes over time. Finally, we validate the performance on a object classification task, by training a classifier on events generated with the Spike-ESIM network.

3.1 Visual inspection

Visualizing the generated events reveals a strong dependency of the method on background noise present in most of the scenes of the HS-ERGB dataset. The amount of noise that is modeled in the scene is strongly dependent on the membrane threshold that we set before training, and its value is extremely important as it strikes a balance between how much noise we allow in the results and how well we want to model the real events.

However, if we ignore the noisy spikes in the background and we focus only on the meaningful visual appearances resulting from the dynamics of the scene, it is possible to observe that the way events are generated is substantially accurate. In Figure 3.1 we show two sequences from the HS-ERGB dataset and present the result of the events generated with Spike-ESIM trained on the real events.

In Figure 3.2, instead, we present the result of training the Spike-ESIM network on the Vimeo-90k-denoising dataset. The sequence shown is taken from the Caltech101 dataset [2]. It is possible to observe that the events generated with the network are almost identical to the real ones.

3.2 Event rates

To evaluate the quality of the generated events from a more quantitative perspective, we focus our attention on the event rates. Comparing how the number

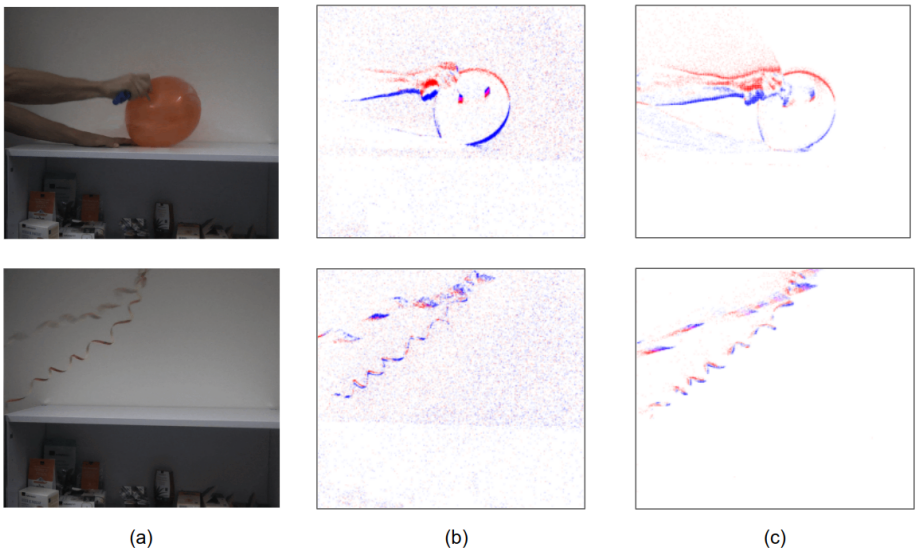


Figure 3.1: A side-by-side comparison of samples from HS-ERGB dataset (a) Input frames (b) Synthetic events generated with Spike-ESIM trained on HS-ERGB (c) Real events.

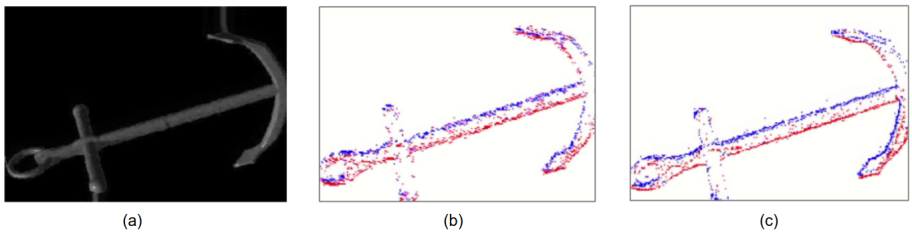


Figure 3.2: A side-by-side comparison of a sample from Caltech101 dataset (a) Input frame (b) Synthetic events generated with Spike-ESIM trained on Vimeo-90k-denoising (c) Synthetic events generated with Vid2E.

of spikes changes over time in both the synthetic and real events can be a reasonable approach to validate the results and to spot potential inaccuracies.

The results of Figure 3.3 show the total event rate of the events generated with Spike-ESIM trained on the HS-ERGB dataset. The sequence taken as reference for the following comparisons is the *baloon* sequence from the HS-ERGB dataset. As one could expect from the results already observed in Section 3.1, the modeled noise causes the event rate to have significant fluctuations because the total number of spikes takes into account also all those spikes related to noise. For a more clear evaluation, a filtering step is performed on the generated events to filter out the noisy spikes. A 5×5 median filter was applied to the resulting event tensor.

In Figure 3.4 we compare the event rate of the events generated with Vid2E, the ones generated with spike-ESIM trained on HS-ERGB and the real events. For

each case, two separate signals are displayed to account for both positive and negative event rates. From the comparison, it is possible to observe that both events generated with Vid2E and spike-ESIM present a flickering behaviour. However, while the events generated with Vid2E tend to overestimate the overall number of spikes by almost double the value of the real events, the method proposed in this work slightly underestimate the real events, although a high level of realism is achieved by capturing most of the visual appearances of the real event stream.

An additional inspection is performed more locally, by taking into account 8×8 patches randomly sampled in the event tensor. Even though the event rate of the generated events does not match perfectly the real events spikes, it still provides a trustworthy reconstruction also locally. From the results of Figure 3.5, we can observe that the two signals are temporally consistent, with no delay introduced in the event generating process, and that their shapes are rather similar.

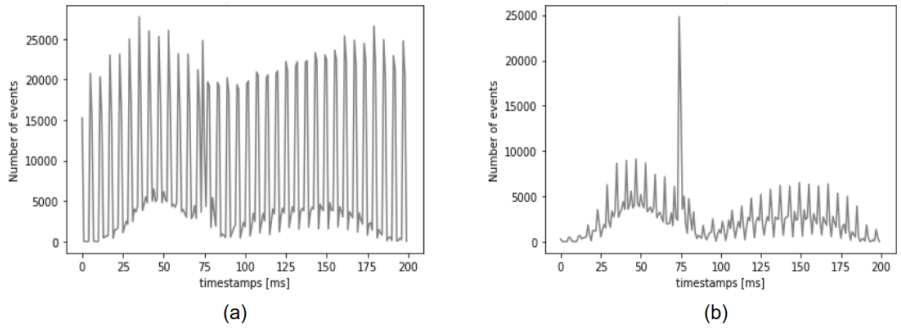


Figure 3.3: A side-by-side comparison of the total number of spikes from the *baloon* sequence of the HS-ERGB dataset, using Spike-ESIM trained on HS-ERGB (a) Total event rate from unfiltered generated events (b) Total event rate after a 5×5 median filtering of the generated events.

3.2.1 Temporal smoothing network

To address the flickering behaviour of the generated events, a new architecture of the network is introduced, performing a temporal smoothing of the events along the positive and negative channels separately. The resulting network, which we called Temporal Smoothing Spike-ESIM, adds a PSP layer and an additional spike function module in each branch of the standard Spike-ESIM network.

Results presented in Figure 3.7 show that the event rate signal resulting from the events generated with this novel network is way smoother than the standard case. In addition to that, it can be observed that the smoothing procedure introduces some delay in the generated events, as the signal is shifted to the right along the temporal dimension by a certain amount. However, from a visual inspection of the generated events with this network shows that we loose some accuracy in modeling the real events.

Although this direction was not further investigated due to lack of time, it could be an interesting starting point for future work to address this problem.

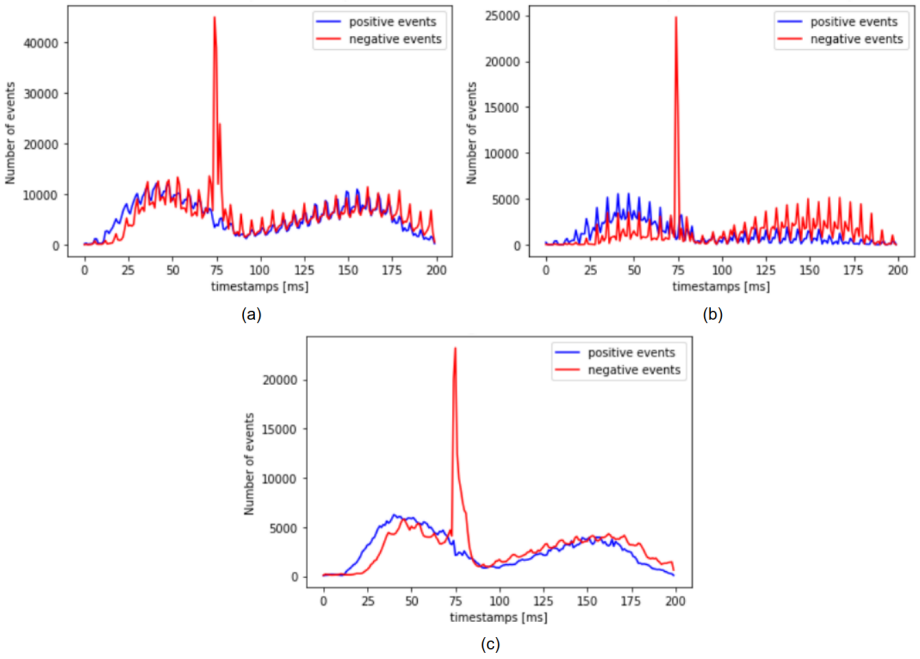


Figure 3.4: Comparison between different methods on the positive and negative event rate from the *baloon* sequence of the HS-ERGB dataset (a) Positive and negative event rate of events generated with Vid2E (b) Positive and negative event rate of events generated with Spike-ESIM trained on HS-ERGB dataset (c) Positive and negative event rate of real events from the HS-ERGB dataset.

3.3 Classification on N-Caltech101

To further investigate the validity of the generated events with respect to other methods, we validate the results on a classification task. We generate synthetic events from the video sequences of the Caltech101 [2], thus creating a simulated replica of the N-Caltech101 dataset [10] (Neuromorphic-Caltech101). We then want to quantify how well a network trained on the simulated N-Caltech101 generalizes to events in the real dataset, NCaltech101. Two versions of the simulated N-Caltech101 dataset are generated, one with an SNN trained on the HS-ERGB dataset and one with an SNN trained on Vimeo-90k-denoising. For the sake of conciseness, we use the abbreviations Spike-ESIM-HSergb and Spike-ESIM-Vimeo to refer to these two networks.

Once these new datasets have been generated, a classifier is trained on top of them separately. The classifier is composed of a Quantization network that first converts the input stream of events into a voxel grid representation, followed by a ResNet-34 [6], which has been pretrained on RGB images from ImageNet [1]. The training is performed choosing a batch size of 4, a learning rate of 10^{-6} and 30 epochs, which are sufficient for the network to converge. The test score is computed on the whole N-Caltech101 dataset, which has been collapsed into one large set, and no more divided into training, validation and test sets. As a baseline we compare the results against a network which was

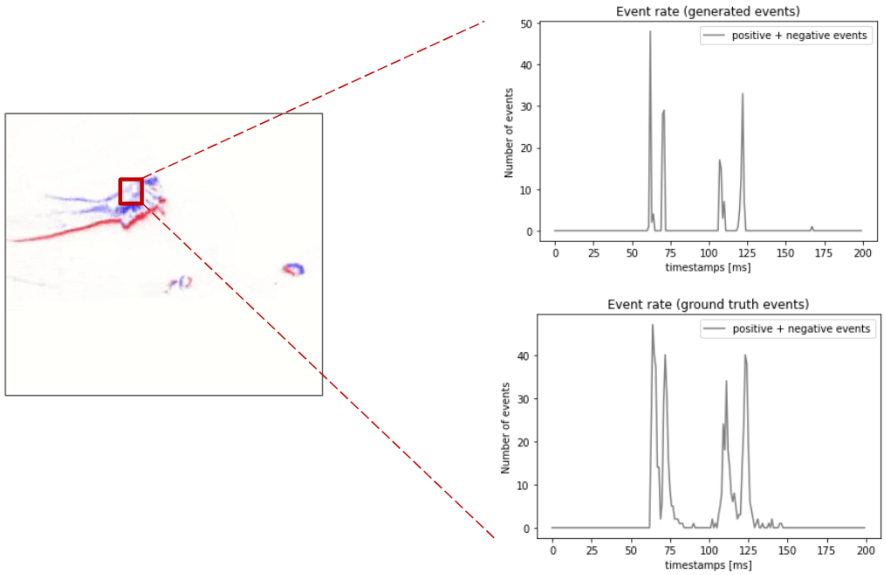


Figure 3.5: Local comparison of a 8×8 patch randomly sampled from the *baloon* sequence of the HS-ERGB dataset on the total event rate between events generated with Spike-ESIM trained on HS-ERGB (up) and ground truth events (down).

trained on synthetic events generated with Vid2E [**?**] and evaluated on the same sequences of the N-Caltech101. From the training results we can observe that the network trained on synthetic events generated with Spike-ESIM-Vimeo with a membrane threshold of 0.7 leads to a higher score (76.2%) with respect to the baseline (75.1%) resulting in an increase of 1.1% in the performance. On the other hand the network trained on synthetic events generated with Spike-ESIM-HSergb leads to a significant decrease in the test score, dropping to 71.5% when using a membrane threshold of 1.5, 70.5% with a threshold of 0.7 and 68.3% with a threshold of 3. The main reasons why this happens might be the fact that the Spike-ESIM-HSergb network has been trained on a significantly noisy dataset and this may affect the overall performance. The same does not apply for the Vimeo-90k-denoising dataset, which has similar characteristics to the Caltech101 dataset, i.e. low background noise and linear motions in the scene.

Method	**Membrane threshold**	**test score**
Spike-ESIM-Vimeo	0.7	**0.762**
Vid2E	✗	0.751
Spike-ESIM-HSergb	1	0.715
Spike-ESIM-HSergb	0.7	0.705
Spike-ESIM-HSergb	3	0.683

Table 3.1: Comparison of classification accuracy on N-Caltech101 using different synthetic datasets for the training and different membrane threshold values.

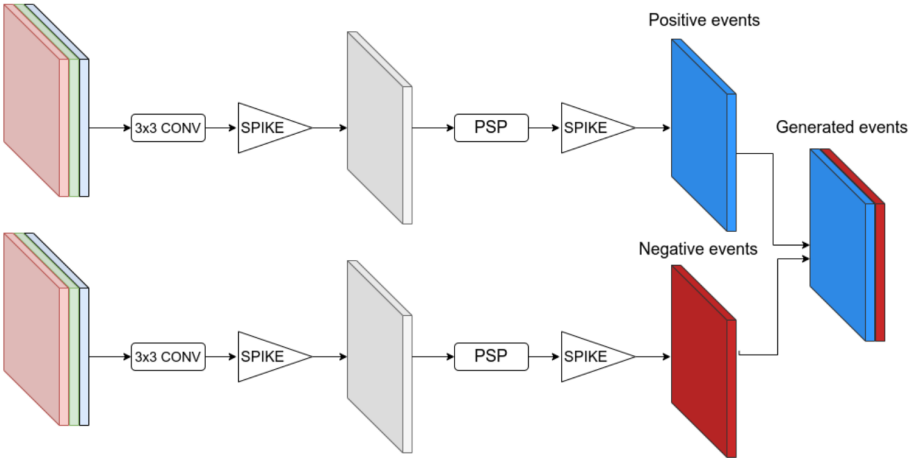


Figure 3.6: Architecture of the temporal smoothing Spike-ESIM network.

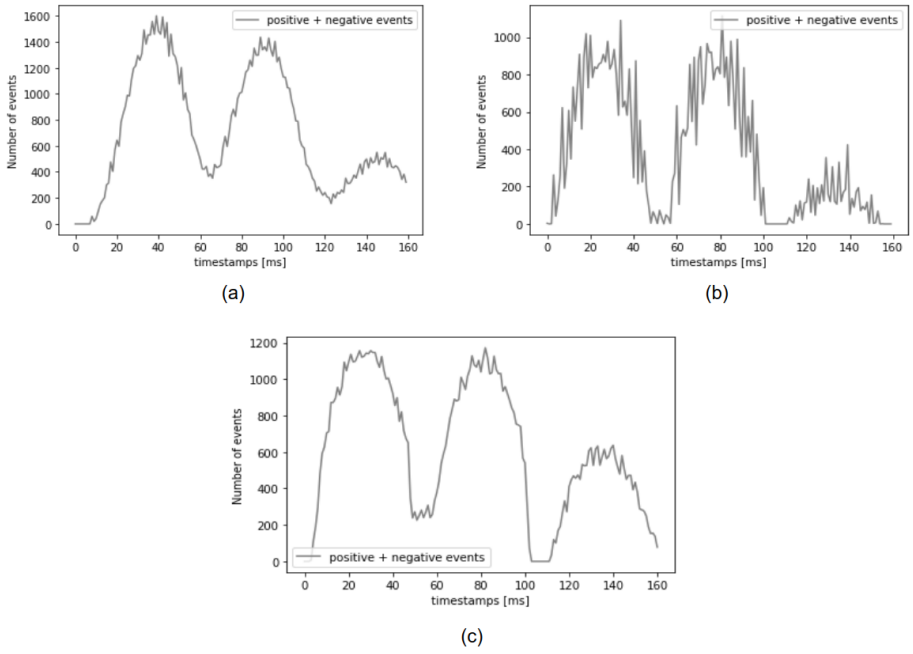


Figure 3.7: Effects of the temporal smoothing network on the *anchor* sequence from the Caltech101 dataset (a) Total event rate of events generated with the temporal smoothing network trained on the Vimeo-90k-denoising dataset, (b) Total event rate of events generated with Spike-ESIM trained on the Vimeo-90k-denoising dataset, (c) Total event rate from ground truth events.

Chapter 4

Discussion

In this chapter we discuss the main achievements of this work, as well as the limitations that were encountered during the process. This work offers a simple, yet effective solution to the problem of lacking event data for computer vision applications, proposing a method for converting video datasets into synthetic event datasets. Although the proposed method achieves better results with respect to previous related methods, there are still numerous limitations that need to be addressed to possibly improve the performance of the model.

4.1 Future Work

Future points of work should focus on the main limitations of the proposed method. First of all, further pre-processing of the input data should be done to reduce the dependency of the method from noise. As the vast majority of datasets are inevitably filled with noise, this represents a crucial step to be undertaken, as we have seen that it significantly affects the performance of the results. Filtering techniques must be deployed prior to the training of the network.

The current implementation of the network is restricted to a small number of trainable parameters during the training process, as we are using single 3×3 3D convolutions for each branch of the network, resulting in a total of 27 parameters per branch. Future work could go in the direction of exploring deeper architectures of the event generation network with more trainable parameters, and possibly integrating with skip-connections.

Moreover, one could test the effect of different loss functions on the training process. One of the main aspects of the events generated with Spike-ESIM is that they generally tend to underestimate the actual number of spikes of the real events. One could design customized loss function to be integrated with the current one, to constraint the learning process directly on the number of generated spikes per sequence.

The results of the classification task on the N-Caltech101 dataset have shown that training on real events does not generalize well on synthetic datasets. To

address this issue, another possible direction that can be developed in future work is to bridge this gap between different datasets to provide a more generalizable setup of the method. An ideal setup would be the following: depending on the dataset we may need to use, one could adjust at test time some hyperparameters that account for these differences.

Finally, to have a better understanding about the validity of the results, one may want to validate the proposed method also on other computer vision tasks, i.e. semantic segmentation, to investigate further advantages or limitations of this work.

4.2 Conclusion

Scarce resources of event data is still one of the major limiting factor of event-based computer vision applications. Prior work has already moved in the direction of leveraging the massive amount of video datasets collected over the years to generate synthetic events, already achieving outstanding results.

We have presented a learning-based solution to generate events from frame-based datasets relying on Spiking Neural Networks. From a visual evaluation of the results, we can say that the synthetic events generated with our method effectively capture most of the visual details of the real event stream and thus they achieve a high level of realism. Furthermore, we highlight the improved performance of the object classification task trained on events generated with this method with respect to previous methods.

There remain many limitations of our work. The proposed method is extremely sensitive to noise in the input video frames and it still lacks of robust generalization capabilities across different types of datasets, limiting its application window to specific cases.

Nevertheless, this project lays the ground work for future research into this direction. It also serves to demonstrate the potential of these exciting and potentially fruitful research avenues for the research community at large.

Bibliography

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[2] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Pattern Recognition Workshop*, 2004.

[3] Daniel Gehrig, Mathias Gehrig, Javier Hidalgo-Carrió, and Davide Scaramuzza. Video to events: Recycling video datasets for event cameras. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, June 2020.

[4] 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)*, October 2019.

[5] Mathias Gehrig, Sumit Bam Shrestha, Daniel Mouritzen, and Davide Scaramuzza. Event-based angular velocity regression with spiking networks. *CoRR*, abs/2003.02790, 2020.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[7] Huaizu Jiang, Deqing Sun, V. Jampani, Ming-Hsuan Yang, Erik G. Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9000–9008, 2018.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[9] Chankyu Lee, Adarsh Kosta, Alex Zihao Zhu, Kenneth Chaney, Kostas Daniilidis, and Kaushik Roy. Spike-flownet: Event-based optical flow estimation with energy-efficient hybrid neural networks. In *European Conference on Computer Vision*, pages 366–382. Springer, 2020.

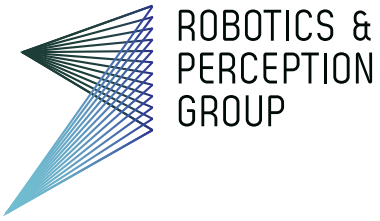
[10] Garrick Orchard, Ajinkya Jayawant, Gregory K. Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in Neuroscience*, 9, 2015.

[11] Henri Rebecq, Daniel Gehrig, and Davide Scaramuzza. Esim: an open event camera simulator. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 969–982. PMLR, 29–31 Oct 2018.

[12] Sumit Bam Shrestha and Garrick Orchard. SLAYER: Spike layer error reassignment in time. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1419–1428. Curran Associates, Inc., 2018.

[13] Stepan Tulyakov, Daniel Gehrig, Stamatios Georgoulis, Julius Erbach, Mathias Gehrig, Yuanyou Li, and Davide Scaramuzza. TimeLens: Event-based video frame interpolation. *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[14] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T. Freeman. Video enhancement with task-oriented flow. *Int. J. Comput. Vision*, 127(8):1106â1125, aug 2019.



Title of work:

Learning to Generate Events using Spiking Neural Networks

Thesis type and date:

Semester Thesis, Jan 2022

Supervision:

Daniel Gehrig
Mathias Gehrig
Prof. Dr. Davide Scaramuzza

Student:

Name:	Ivan Alberico
E-mail:	ialberico@student.ethz.ch
Legi-Nr.:	20-948-832

Statement regarding plagiarism:

By signing this statement, I affirm that I have read the information notice on plagiarism, independently produced this paper, and adhered to the general practice of source citation in this subject-area.

Information notice on plagiarism:

`http://www.lehre.uzh.ch/plagiate/20110314_LK_Plagiarism.pdf`

Zurich, 30. 1. 2022: