Using Spiking Neural Networks to Estimate Depth from Stereo Neuromorphic Events

Mid-Term Report

Ulysse Rançon

Supervisors: Benoît Cottereau
Timothée Masquelier

Department of Electrical Engineering
INSA de Lyon

This dissertation is submitted for the degree of
Ingénieur

INSA Lyon - Cerco CNRS UMR5549 May 2021

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Ulysse Rançon
May 2021

Acknowledgements

I would like to deeply thank my supervisors Benoît and Timothée, without whom I would not have been able to realize such an interesting master's thesis. Thank you so much for trusting me and welcoming me when I applied for this position at the Cerco on my own. To be quite honest, this was one of my dream internship, and I am so happy to have had the chance of working with you. Thank you also for teaching me so much about the job of the science researcher in general.

My thanks also go to my fellow intern Javier, with whom I had the pleasure to work during the week and play board games during the weekend.

I would also like to express my gratitude to the whole Cerco team for their warm welcome in spite of the difficult conditions due to the covid-19 pandemic.

Of course, I have to thank the staff at Kohno laboratory at the University of Tokyo, because it is where my passion for neuroscience all started. Similarly, I am grateful towards the people I met during my previous internship, at the Computer Vision lab at Bull in Echirolles. I learned so much about deep learning, programming and IT tools there; it saved me a lot of precious time to do this research work.

Finally, I would like to thank my family, and especially my grand-parents, uncle and aunt, who supported me during this intensive period in Toulouse.

Abstract

Spiking neural networks (SNN) are getting more and more interest in the AI community, as they use temporal information, to the contrary of their "regular" ANN cousins. They could even outperform the latter in the coming years and become a new paradigm for complex computation. Indeed, it is already broadly recognized that their hardware implementation on neuromorphic chips is far less energy consuming than classical GPUs. In this context, a growing number of innovators and constructors propose compatible hardware, especially in the field of vision. Dynamic Vision Sensors (DVS) capture the dynamics of a scene and moving objects with inspiration from the retina. Instead of taking shots at regular intervals, it rather produces spikes in extremely high temporal resolution (in the order of the microsecond), whenever a detected luminance change in a given location of the field exceeds a threshold. They are now being investigated for countless applications and at different levels, such as object detection, embedded systems (edge AI), or videosurveillance.

This project aims at developing a SNN that estimates the depth of each pixel in a scene, given two separate spike trains obtained in a bio-inspired manner with DVSs.

In addition to providing a new solution for a practical engineering task, this work could lead to modest but still appreciable insights on the functioning of the Visual Nervous System.

Table of contents

Nomenclature

Biology

BEM	Binocular Energy Model
CNS	Central Nervous System
EEG	Electroencephalogram
IT	Inferior Temporal Cortex
LGN	Lateral Geniculate Nucleus
MT	Middle-Temporal Visual Area
PFC	Prefrontal Cortex
RDS	Random Dot Stereogram
V1, V2	Primary, Secondary Visual Cortex

Deep Learning

AI	Artificial Intelligence
ANN	Analog Neural Network
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
SNN	Spiking Neural Network
(T)BPTT	(Truncated) Back-Propagation Through Time

Neuromorphic Hardware

AER	Address Event Representation
DVS	Dynamic Vision Sensor

Spiking Neurons

IF	Integrate-And-Fire
LIF	Leaky Integrate-and-Fire
LTP/LTD	Long Term Potentiation/Depression
PLIF	Parametric Leaky Integrate-and-Fire
SG	Surrogate Gradient
STDP	Spike-Timing-Dependent Plasticity

Other Symbols

MDE	Mean Depth Error (unit: m)
spk	spike, action potential, event

Chapter 1

Introduction

1.1 Personal Motivations

My interest for neuroscience originally traces back to my exchange studies at the University of Tokyo, Japan. There, I integrated a laboratory[1] aiming at implementing *in silico* biomimetic neurons, following complex, highly non-linear mathematical models.

Greatly excited by this research topic, I grabbed this rare opportunity to discover the world of academic research by practical problem solving. To be noted is that the way of studying in Japan is indeed very different from the French model of engineering schools. In a semi-supervised manner, I played simulating various neuron models equipped with Spike-Timing Dependent Plasticity (STDP) learning rule and tried to replicate an influential paper by Timothée Masquelier[14], actually the first scientific research article that I read. This experience was a revelation to me: I had to do a PhD and a research career in the field of computational neuroscience.

Equally excited for artificial intelligence (AI), this conviction led me to do my M1 internship in a private R&D team at Atos-Bull in Grenoble, where I built for myself a strong background in deep learning. More importantly, this previous internship allowed me to learn how to use many common programming and computer tools, that are the bread-and-butter for any machine learning researcher.

Then came the time of my master's thesis, validating five years of intense engineering studies. Still willing to continue the shift from electrical engineering to computational neuroscience and AI, I decided to spontaneously apply at the *Centre de Recherche Cerveau et Cognition* (CerCo), whose papers first got me passionate about spiking neural networks (SNN). I cannot thank enough my supervisors Benoît and Timothée for kindly considering

[1] Laboratory for Neuromimetic Systems, Institute of Industrial Science (IIS), The University of Tokyo, https://www.neumis.iis.u-tokyo.ac.jp/

my application, trusting me and welcoming me among their respective teams.

I guess that French engineering studies are general enough to lead to many points; this personal opinion is comforted by the fact that both of my supervisors initially come from an engineering background, as it will be further exposed. I am happy to have renewed with my long-lasting interest for biology, while being able to use neuroscience to the service of AI and AI to the service of neuroscience. I am proud of my specialty which allows me, in the end, to join in my own way that of my parents, both psychiatrists. The upcoming next stage in my career will be a PhD thesis implementing a SNN on FPGA for spike-sorting real eletrophysiological data, in the context of a neural implant. This internship could not have been a more valuable preparation for this task, and I feel ready to and look forward doing more research on the intricate magic of the brain.

1.2 Host Organization

This master's thesis took place at the *Centre de Recherche Cerveau et Cognition*[2] or *Brain and Cognition Research Center* (CerCo) in Toulouse, France.

Located inside the Purpan *Centre Hospitalier Universitaire* (CHU), this laboratory is a joint CNRS/Paul Sabatier University unit, and focuses on the study of different sensory modalities and their integration, cognitive functions such as memory, object recognition but also consciousness and mental states. Grouping a staff of nearly 70 full-time researchers, postdocs, PhD and master students, it is divided into 5 teams having different research themes, two of which I integrated during the course of my internship.

The SV3M team -standing for *Spatial Vision in Man, Monkey, and Machine*- brings together researchers and clinicians with complementary expertise who aim at better understanding the neural and cognitive mechanisms underlying spatial vision. Together, they have developed an innovative approach which is resolutely multi-model (healthy humans, patients, monkey and machine) and trans-disciplinary (behaviour, eye tracking, electroencephalography, cortical stimulation, functional imaging in human and non-human primates, modelling). The team's current project explores spatial vision under naturalistic conditions, focusing on its roles in navigation (axe I), on the mechanisms by which it acquires selectivity, expertise and plasticity (axe II) and on its interactions with other cognitive functions (axe III). Additionally, these three axes feed a fourth axis of translational research with strong anchors

[2]http://cerco.cnrs.fr/en/cerco-umr5549-2/

in both clinical and technological domains. Part of the team are also students enrolled in a neuropsychology master at the Toulouse III Paul Sabatier University.

My first supervisor, Benoît Cottereau, currently co-directs this team. Graduated in signal processing from the French school of engineering Centrale Supelec, and having received a PhD degree from University of Paris XI on EEG and MEG signals, his preferred research subjects gather brain imaging (EEG, MEG & fMRI) and 3D encoding in the human and non-human primate visual cortex (disparity processing, spatial coordinate transformation and depth perception).

On the other hand, the neuroAI team focuses on more theoretical and abstract aspects, and less on experimental biology. It aims to make a bridge between neuroscience and AI. Namely, members use state-of-the-art algorithms to address important neuroscience questions. For example, deep learning tools are used to find patterns in vast amounts of fMRI and EEG data, and to relate them to stimuli, cognition, behavior and well being. Conversely, they use knowledge of the brain to design improved algorithms and artificial neural network (ANN) architectures, for example by incorporating spikes, feedback, oscillations, or more human-like representations. PhD and master students in this team rather come from a mathematics or engineering background.

Timothée Masquelier, my second supervisor, is co-director of this group. Graduated from Supelec as well, he pursued his studies at the MIT and received a PhD at the University Toulouse III. As a researcher in computational neuroscience and AI, he uses numerical simulations and analytical calculations to gain understanding on how the brain works, and more specifically on how neurons process, encode and transmit information through action potentials (a.k.a spikes), in particular in the visual and auditory modalities. Also interested in bio-inspired computer vision and audition, he is convinced that spiking neural networks (SNN) can be an appealing alternative to the conventional artificial neural networks commonly used in AI, especially if implemented on low-power neuromorphic chips.

Finally, I cannot omit my fellow intern Javier Cuadrado-Anibarro, an M2 student from ISAE-SUPAERO, who joined me a month after my arrival at the CerCo for a similar master's thesis. While I have been developing SNNs for depth estimation from neuromorphic data, Javier has adopted the same approach, but for optical flow estimation instead. Because of the closeness of our subjects, we have helped each other a lot and organized weekly meetings together with our supervisors.

1.3 Workspace

The subject of this master's thesis being abstract and quite theoretical, it did not require much hardware materials. I mainly worked on my personal computer, equipped with a small GPU for program developing and testing (Nvidia GeForce GTX 1650, 4GiB VRAM). Large-scale tests and trainings of models were made on a distant GPU owned by the laboratory, accessible *via* SSH protocol: an Nvidia Quadro RTX 8000 with 48GiB VRAM.

As will be explained later in this report, memory is an important limitation in training SNNs, and a key training factor for boosting the performances of models. Hopefully, 48GiB of GPU memory seem sufficient for efficient training.

Programming was done in Python, using *spikingjelly* library[5], an open-source deep learning framework for developing SNNs based on PyTorch. Due to its high compatibility with the latter and its easy syntax, it is considered a rising star in the SNN community and could soon rival other important frameworks. Additionally, it features unique behind-the-kitchen CUDA kernels, greatly speeding up computations compared to naive PyTorch, while still benefiting from its high-level syntaxic simplicity. For instance, we reported up to 50% training speed increase on certain toy trainings.

Finally, codes were regularly updated on a private Github repository, which will be made publicly available at the end of the internship, or after potential publications. This allowed my tutors and my fellow intern to easily see the state of my work and its updates. Similarly, I could follow changes on his work by checking his own repository[3]. Mine is available at https://github.com/urancon/Depth-SNN.

[3] https://github.com/J-Cuadrado/SNN-OpticFlow

Chapter 2

Subject Presentation

2.1 Depth Processing in the Brain

The visual nervous system represents a major part of the processing done in the animal brain. For primates, vision is estimated to constitute as much as 30% of all computations occurring in the central nervous system. It solves many high-level tasks in parallel, such as object recognition in the ventral visual pathway, or spatial cognition in the dorsal visual pathway. Three-dimensional (3D) navigation itself is a complex process, relying on a variety of visual information, such as optical flow or depth.

Binocular disparity is the slight difference in the retinal images between both eyes and is one of the primary cues for depth perception in a wide range of animal species. Monocular depth cues also exist, such as light intensity, chromaticity, perspective, shape from shading or motion parallax [2]. We distinguish two types of signal for the stereo disparity systems in the brain: *absolute* and *relative disparity*. Supposing that an eye E stares at two points P and Q in the field of view, the absolute disparity is the angle between all three points. The relative disparity between two points is the difference between their absolute disparity; taking this difference eliminates the fovea as a reference point. Relative disparity is therefore independent of ocular fixation. See figure below for some visual help.

While it has been a long standing problem, binocular stereopsis is still poorly understood today. However, we know that as early as the first cortical area of the visual system (V1), neurons are tuned to binocular disparity. Some neurons indeed combine signals from both eyes, before projecting to deep layers of the visual nervous system. Roughly, the shallower the neurons in the visual nervous system (i.e., the closer they are to the sensory neurons of the retinae), the smaller their receptive fields. Deeper neurons gradually become receptive to larger spatial dependencies, and encode higher-level features. For instance, early features could be edges, while later features could be geometric shapes such as squares or circles.

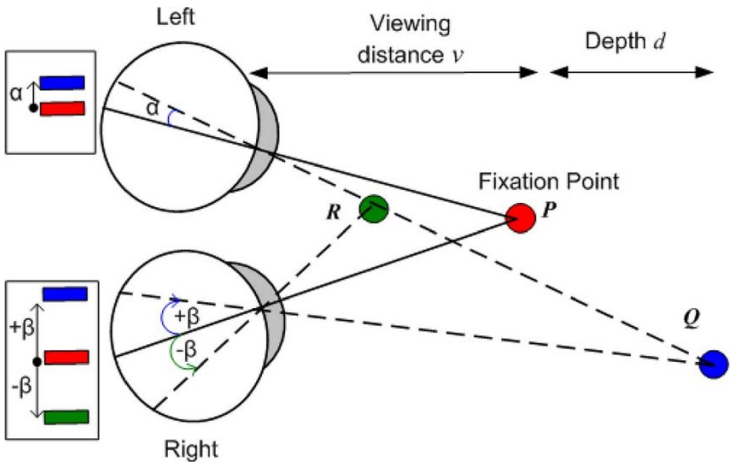

Fig. 2.1 Geometric diagram of stereopsis, borrowed from "Display Dependent Preprocessing of Depth Maps Based on Just Noticeable Depth Difference Modeling", Da Silva et al. (2011). Absolute disparities are β and α, while relative disparity is $\beta - \alpha$.

2.2 Dynamic Vision Sensors

Because of the remarkable efficiency of the brain in vision tasks at a minimal energy cost (around 10-25 W), engineers have started to take inspiration from its way of processing sensory stimuli and make complex computations. Misha Mahowald was a pioneer and first proposed a neuromorphic sensor, the *silicon retina*. It mimicks the functioning of rod cells, cone cells, and retinal ganglion cells, present at the first layers of the retina.

Also referred to as "event camera" or "dynamic vision sensor" (DVS), this class of sensors is radically different from classical frame based cameras, which capture and restitute the luminance in the field of view at fixed time intervals (say, 20ms). Instead, pixels are independent from each other and asynchronously emit an action potential, or *spike*, or *event*, whenever the log-luminance of the point in the scene they aim at, exceeds a threshold. More formally, a pixel at position (x,y) will elicit a spike at time t if:

$$||log(I_{x,y,t}) - log(I_{x,y,t-\Delta t})|| \geqslant \theta \tag{2.1}$$

Δt, which defines the temporal resolution of the system, is typically in an order of magnitude of 10 µs. Naturally, this detected change in log-luminance can either be positive (denoting an increase) or negative (decrease); this leads to a polarity property of the emitted action potential, which can be either an *ON event* or *OFF event* respectively.

Therefore, the format of the data coming out of a DVS much differs from traditional channelled and frame-based representations (e.g., RGB, HSV). Instead, a common format is the address event representation (AER), which is no more than a list of events in chronological

order, each described by a quadruplet of features: x position, y position, polarity, and timestamp.

Fig. 2.2 **Left:** the DAVIS m346B event camera, used for the acquisition of the MVSEC dataset. It has a resolution of 260x346 pixels, and costs around 4,500€. Source: https://inivation.com **Right:** the drone rig used to shoot MVSEC sequences. One can recognize the two white DAVIS on its the front. Borrowed from [25].

Because of these sensing properties, event cameras essentially capture the rapid changes (e.g. motion) in a scene. Intuitively, no spike will originate from a fully static scene, while traditional cameras will repeat the capture of the luminance frame, even though it is redundant. As a result, DVSs encode information in a highly optimal way.

There are several strong motivations for using DVSs. A first one is their very high temporal resolution, whose order of magnitude can be thousands of times lower than classical frame-based cameras (typically about 10 µs). Second, they benefit from a very high dynamic range of around 120 dB, compared to an average value of around 50 dB for regular cameras, allowing them to capture scenes under very bright or very dim light conditions. They are also subject to significantly less motion blur than frame-based systems. Finally, as the consume power only at the emission of spikes, which are rather sparse, they are highly energy-efficient with a power consumption not greater than a few tens of milliWatts. All these advantages makes them sensors of choice for a wide range of automotive applications, embedded systems, but also surveillance cameras.

Despite these perks, DVS suffer some drawbacks that their frame-based relatives do not have. Probably the most important is their low spatial resolution, which rarely attains 1 megapixel. They also do not restitute color information, contrarily to the luminance fields measured by classical sensors. Moreover, they remain still quite expensive (several thousands of euros); however their price may decrease in the coming years because of their attractive traits and rising popularity. Another disadvantage is the lack of computer vision algorithms to process events. To fill this gap is precisely the purpose of this internship.

2.3 The MVSEC dataset

The Multi Vehicle Stereo Event Camera (MVSEC) dataset [25] is a collection of neuromorphic data acquired with two side-by-side DAVIS m346B event cameras mounted on a moving rig (car, motorbike, hexacopter) and in different light conditions (day, night). In addition, the depth of the scene was obtained in parallel thanks to a Velodyne Puck LITE lidar. The latter has a sampling frequency of 20 Hz, hence producing depth images at intervals of 50 ms. This gives us two streams of spikes and groudtruth depth maps that can serve as inputs and labels for a supervised learning approach. MVSEC is the first dataset to propose stereo event data accompanied with groundtruth on a moving scenario, and has been gaining some popularity over the last two years. It is versatile and is not limited to depth for groundtruth: as an example, it provides optical flow groundtruth, as well as GPS and motion capture acquisitions.

2.4 Spiking Neural Networks

Spiking Neural Networks (SNN) are to the computing units of the visual nervous system (and more generally, the animal brain) what the silicon retina is to its visual sensory units. Sometimes referred to as "third generation Artificial Neural Networks", they are more biologically plausible counterparts of the latter, and promise a bright future.

Although Analog Neural Networks (ANN) already are inspired from the brain, SNNs make one step forward towards biological plausibility. Indeed, SNNs share some properties with biological neural networks, such as binary outputs and sparse activity.

Essentially, each unit of a SNN represents an actual biological neuron with a simplified mathematical model consisting of a system of nonlinear differential equations. Arguably the simplest model is the leaky integrate-and-fire model (LIF), first proposed by French neuroscientist Louis Lapique as early as in 1907. He noticed that a neuron is similar to an excitable RC circuit with a threshold over which an action potential is emitted. It can be expressed as:

$$LIF\ model: \begin{cases} \tau_{mem}\frac{dU_i^{(l)}(t)}{dt} = -(U_i^{(l)}(t) - U_{rest}) + RI_i^{(l)}(t) & \\ U_i^{(l)}(t) = U_{rest} & \text{if } U_i^{(l)}(t) > U_{thresh} \end{cases} \tag{2.2}$$

There are many other, much more complex and realistic neuron models, that we will not present in this report for the sake of conciseness. Perhaps we can at least enumerate the exponential integrate-and-fire model, the spike response model (SRM), the Izhikevich

model (IZH), or the Hodgkin-Huxley (HH) model. Important to note is that the more complex the model, the more different behaviours a neuron can take, and the closer it can be to the reality of biological neurons. In theory, an increased model complexity could be followed by a better learning power in a machine learning (ML) setting. In practice however, it is not feasible to use too complex models for developing SNNs, because of the great amounts of computation they require, impossible to carry with current computers. This is why most SNNs currently proposed by the deep learning community revolve around the LIF model, which captures some essential properties at a relatively reasonable computational cost.

Because of their formulation as a set of variables bound in time by a set of differential equations, spiking neurons are *stateful* units. Indeed, in the case of the LIF neuron, the leaky membrane potential plays a role of short-term memory. Its integration of spike trains coupled with its firing mechanism allows it to behave as a coincidence detector.

Because of their binary nature and their internal state, one can see SNNs as a particular case of RNNs [16].

Strong properties motivate the use of SNNs rather than their analog counterpart. Firstly, they are stateful in time, enabling them to retrieve one supplementary dimension in the data: time. Secondly, the binary output of its units coupled to lightweight computation and an overall sparsity of spikes accross the networks allow for extremely energy-efficient hardware implementations on dedicated hardware, such as neuromorphic chips, or silicon integrated circuits (SiC), or field-programmable gate arrays (FPGA). In comparison, ANNs running on GPU are typically much more energy consuming.

Among recent and well-known such neural processors, one could cite Intel Loihi[1], IBM TrueNorth[2] or Akida neural processor[3] by Brainchip, which all embark digital spiking neural networks with very low-power consumption. They are configurable in the sense that it is possible to set the connectivity and weights between neural units. With the aim of being implementable on such hardware, this master's thesis aims at determining a performing architecture for the problem of depth estimation. Thus, learning is done *offline* in this context. As of today, the horizon of hardware neuromorphic computing would be the development of working, large-scale architectures able to learn in an *online* manner. As these chips are still undergoing intense research and development, they are not publicly released yet. Finally, we will mention Brainscales[4] and spinnaker[5] systems developed by public European

[1]https://www.intel.com/content/www/us/en/research/neuromorphic-computing.html
[2]https://research.ibm.com/articles/brain-chip.shtml
[3]https://brainchipinc.com/akida-neuromorphic-system-on-chip/
[4]https://brainscales.kip.uni-heidelberg.de/
[5]http://apt.cs.manchester.ac.uk/projects/SpiNNaker/

research institutes, and available for heavy computational neuroscience use. Contrarily to the preceding chips, the latter are not intended for edge AI, embedded applications, but rather research purposes.

2.5 Related Work

Depth estimation is a long standing task in the computer vision community and has first been investigated with success by deep learning techniques in [8]. It used traditional frame images for training in an unsupervised learning fashion, and inspired many other papers. After its success at depth estimation on classical frame data, the scientific community has recently turned its focus towards neuromorphic event data, and has tried to tackle this task on the MVSEC dataset.

In [11], an ANN learns to make depth predictions from MVSEC monocular event data, with excellent performances. The main ideas to take away from this paper is a UNet-shaped architecture [20], its use of ConvLSTM for encoding layers only, and its pretraining on a synthetic new dataset with perfect groundtruth based on the CARLA simulator [3]. However, we would like to relativize this success, because it addressed the problem of depth estimation on outdoor data, on which more monocular depth cues are more easily extractable than on indoor data. More precisely, because distances on outdoor scenes are higher, binocular disparity does not allow to estimate depth as well as indoors and monocular information becomes more suitable in this case. Also, their model learns on the logarithm of groundtruth depth maps, which we believe to be more efficient indeed.

[21] is one of the few works to have explored the task of depth estimation from binocular event data with deep learning based methods. If the proposed architecture is still constructed around an encoder-decoder backbone, it is substantially more complex and involves LSTM units, temporal convolutions and a new event sequence embedding. Because we wanted for our own work a more straightforward architecture, we did not retain so many ideas from this article, but it offers nevertheless a challenging baseline for performance comparison on the same stereo setting.

Because optical flow estimation is a task that is very related to depth estimation, we would also like to present some related approaches that inspired us.

The EV-FlowNet model [26] was proposed by the same authors of the MVSEC dataset, and was the first approach to estimate optical flow from monocular event data. Like in the depth estimation approaches, it features a UNet-like, encoder-decoder network architecture. This network consists of non-recurrent (i.e., not stateful) analog neuronal units, and outputs

intermediary predictions at the different scales of the decoder. A multiscale loss is computed with all intermediary maps; the latter are then re-integrated to the main upsampling branch of the decoder for further preprocessing (see figure below). By adding constraints to the predictions at different scales, this structure ensures better convergence to a good set of parameters during optimization.

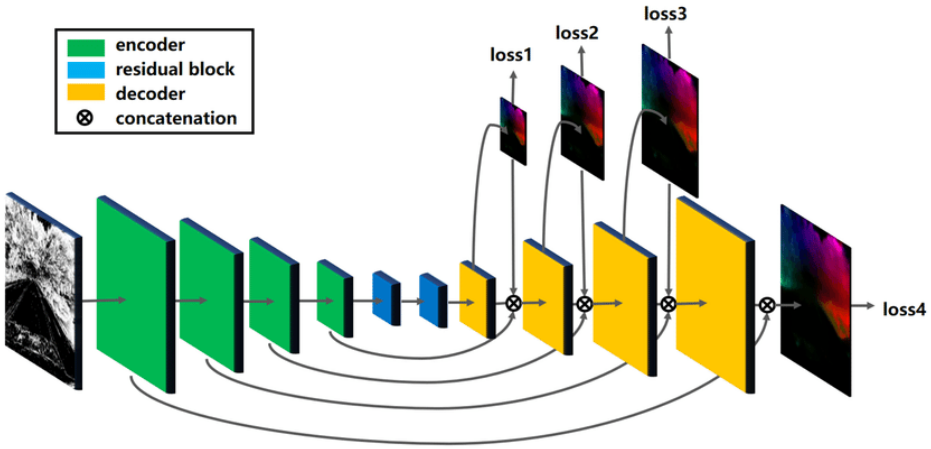

Fig. 2.3 EV-FlowNet architecture, taken from [26]. Shaped like a UNet network, it is composed of an encoder, a bottleneck and a decoder. The encoder applies a first processing to the input spike frame, while gradually decreasing the resolution and increasing the number of channels. The decoder does the opposite: it increases the resolution back to its original dimensions, and decreases the number of channels. Skip connections combine feature maps from the encoder and the decoder at the same level.

Spike-FlowNet [12] is an answer to EV-FlowNet in that it addresses the same problem, but by integrating SNNs at the encoder layers and keeping ANNs for the residual blocks and the decoder. If this model has arguably an increased computational and energy efficiency due to its SNN encoder, the use of ANNs in deeper layers unfortunately undermines this effort. Besides, the overall method is similar, with an architecture close to that of EV-FlowNet and the same multi-scale loss.

Finally, the same authors proposed Fusion-FlowNet [13], another hybrid SNN-ANN architecture that uses one DVS spike train and one grayscale image. Spiking cameras such as the DAVIS m346B that is used in MVSEC dataset can indeed also produce grayscale image frames just like regular sensors, so the authors pragmatically use this extra data at zero additional hardware cost. Moreover it is true that the brain also uses luminance fields as supplementary depth cues [19]. However biological neurons remain spiking, so the SNN-ANN hybridation makes their model not biologically plausible.

2.6 This Work - Our Position

Our approach is innovative in several aspects.

On the side of AI, we propose the first attempt to estimate depth from binocular event data with a fully-spiking neural network. Perhaps even more importantly, it is the first investigation of SNNs for such a large scale regression task. We firmly believe that SNNs are models of choice for processing neuromorphic data, such as streams of spikes from DVSs. Resolutely pushing as far as possible the SNN philosophy, we ideally would like our network to be easily implementable on neuromorphic hardware, for a very low-power depth estimator on an embedded system. Viable depth predictions from such a setup would mean that event cameras encode enough information to be able to replace an expensive Lidar, to a certain extent. This idea could be generalized to other visual measurements. Good performances of our network could translate into a potential publication presenting the originality of our method, and the proof that SNNs can be applied for difficult numerical regression of real values, and not only classification and detection tasks

The neuroscience side has many experimentations to offer as well. After the formulation of a successful neural network, we would like to compare activations of encoding layers with tuning functions of early layers of the visual nervous system, such as areas V1 and V2. The processing of binocular disparity remains not well understood, and such a study could be an valuable contiribution to the field.

Chapter 3

AI: Proposed Algorithm

3.1 Network Architecture

Like many other similar deep learning approaches for depth or optical flow estimation, we adopted a UNet-like architecture for our SNN. More specifically, it is inspired from Fusion-FlowNet[13], but instead of processing a grayscale image, the second encoder takes as an input the stream of spike of the second DVS.

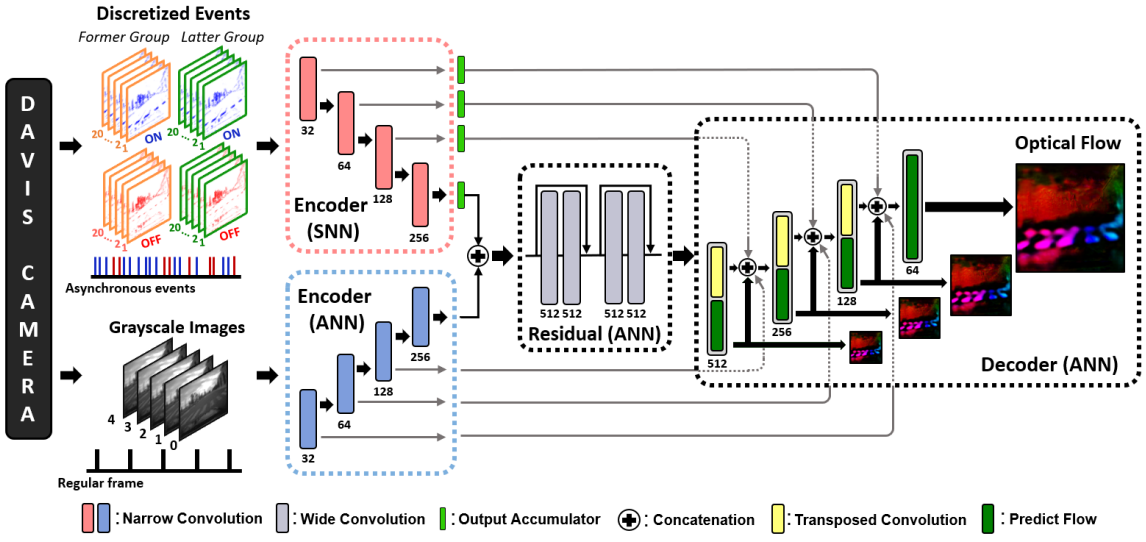

Fig. 3.1 Fusion-FlowNet architecture, as defined in [13]. In the future, when we settle our final network architecture, we will replace this diagram by one of our network. For now, just take note on the overall architecture: 2 encoders whose outputs are concatenated, passed through a bottleneck, and then gradually upsampled with skip connections.

Contrarily to Fusion-FlowNet, our network is fully spiking, meaning that the bottleneck and decoder parts are SNNs too. In this philosophy, we decided that the depth predictions

would be carried by the membrane potentials of output neurons. This encoding of depth could easily be implemented on a neuromorphic chip. Moreover, these output neurons are non-leaky and do not spike (i.e., their spiking threshold is infinite). Therefore, they are perfect integrators of preceding weighted spike trains and could be called "I (Integrate) neurons"; their only way of increasing or decreasing their output potential (i.e., carrying depth) is by receiving respectively positive or negative weighted spikes from preceding layers. Therefore, in the absence of input spikes -meaning that the observer or the scene are not moving, the depth predicted by our SNN does not change. This is a very important feature that we wanted to transcribe in our architecture.

Another difference with other encoder-decoder architectures is that the combination at the skip connections between downsampling and upsampling layers, is not a concatenation, but rather, a simple addition of spikes.

Furthermore, spiking neurons in our network follow the Leaky-Integrate-and-Fire (LIF) model with a learnable time constant presented in [4], also known as "Parametric LIF" (PLIF). Thus, PLIF layers can find different optimal values for their time constant after training. In addition, time constants being learnable is comforted by the phenomenon of myelin plasticity observed in the biology [1].

The following sections will expose the details of each layer, from the most shallow to the deepest.

3.1.1 SEW Residual Blocks

Back in 2015, the introduction of Resnet [9] was a revolution in the deep learning community. Resblocks allowed the construction of very deep networks not suffering from the vanishing or exploding gradient problem. Until recently, SNNs suffered the same problems as pre-Resnet ANNs: an impossibility to learn with deep architectures and a vanishing of spikes in deeper layers. The introduction of surrogate gradient learning (see section 3.2.1) and more recently of spiking residual blocks[6] allow very deep architectures to avoid this problem and reach yet better performances. For the bottleneck of our network, we use two subsequent Spike Element-Wise (SEW) Resblocks defined in [6], with 3x3 2D convolutions and an additive connection function.

3.1.2 Upsampling Layers

Although it is known that they can produce serious checkerboard artifacts, many of the similar approaches that we mentioned earlier use transposed convolutions (a.k.a. "deconvolution")

layers for upsampling at the level of decoder layers. [17] first noticed the causes of this phenomenon, and suggested using a classical interpolation (e.g. linear, bilinear, bicubic...) followed by a convolution instead to efficiently get rid of these patterns. However, we found commonly used interpolation techniques not compatible with the philosophy of SNNs. Indeed, they can result in a non-integer values in the upsampled spike tensor (e.g. a linear interpolation between a 0 and a 1 would give out a value of 0.5), whose elements should be integer number of spikes at each position. This is neither biologically plausible nor easily implementable on neuromorphic hardware. To avoid the creation of deconvolution artifacts while remaining SNN-friendly, we chose the Nearest Neighbour (NN) interpolation technique followed by a 3x3 2D convolution. We call this kind of layer *NNConvUpsampling*. Because it keeps integer values in its output, NN interpolation is compatible with SNNs. It also requires less computations and is faster than other interpolation techniques. With NNConvUpsampling layers, we report no checkerboards at all and obtain smooth depth predictions.

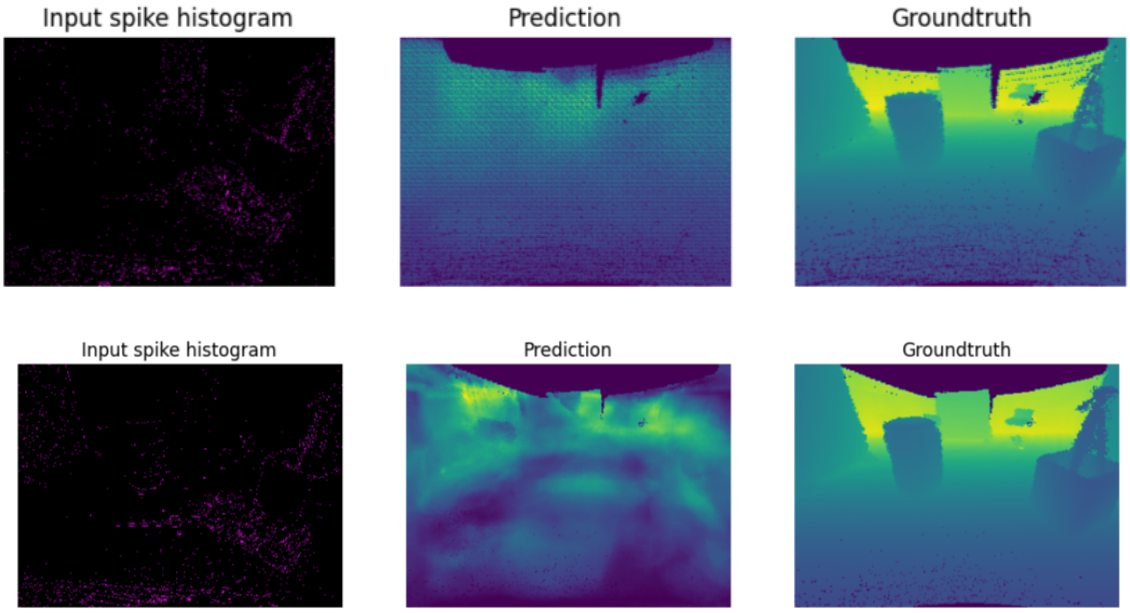

Fig. 3.2 Predictions of our network using transposed convolution layers (top) versus using NNConvUpsampling layers (bottom). Notice that the checkerboard artifacts present in the first case are completely smoothed out in the second case. We recommended printing this page in color and high quality for better appreciation.

3.1.3 Output Layers

The "real" bottleneck of our SNN is actually and without hesitation its output layer. It represents all the difficulty of regressing real numerical values with a network of neurons whose outputs are binary in nature. We have been struggling to find a good architecture to go along our initial postulate of output neurons bearing the predictions of depth with their potentials. Because of this formulation, the potentials of output neurons have the dimension of a length [m]. Therefore, the weights connecting them to the preceding neurons have the dimension of a length over a unit of action potential [m/spk]. For instance, 3 spikes from a preceding neuron with a weight of +1.2 spk/m will change the subsequent output neuron potential by 3.6 m. The problem here is that there cannot be a non-integer number of spikes. Hence, to predict a depth of 4 m at this same output neuron, the network would have to choose to make the same preceding neuron spike 3 times instead of 4 to minimize the error. Considering the example of these only two neurons connected by this synapse with this weight, it is impossible get an error smaller than 0.4. We also remind that output depth potentials should be able to decrease when an object is moving away from the observer. Finally, and to add to the difficulty, the number of spikes that an output neuron receives between two depth predictions is limited, because of memory constraints. Of course, the number of weights between output neurons and the preceding layer is limited as well.

In practice, we noticed that our network lacked expressive power in its output potentials. That is, it especially lacked dynamism and predicted a sort of slowly changing "mean" depth image that would, "on average", minimize its error with all groundtruths of the evaluation sequence. This average prediction would not transcribe well objects that stand out of the visual field, and its update would be on a much slower time scale than the sequence, translating in a lag between the prediction and the sequence.

It is therefore extremely difficult to accurately predict the depth values of 260x346 pixels with SNNs. A success at this task would constitute a true *tour de force* and could inspire the application of SNNs on a number of other regression tasks. We have not yet been able to find a satisfying architecture that solves these problems, and it will be our next priority.

3.1.4 Attention Blocks

Because one major flaw of our SNN was its unability to render objects that stand out of the background, we thought that the incorporation of visual attention could constitute a great asset to our model. Attention has been an active field of research in the deep learning community over the past few years, following the release of the seminal paper [22]. Early

works investigated temporal attention with a particular focus on natural language processing (NLP). Since then, researchers have tried to use similar mechanisms but for image processing and computer vision.

In our case, we wanted to remain close to the classical architecture of a recurrent encoder-decoder, which is more biologically plausible than that of a transformer[22] network. For this reason, we took inspiration from Attention-Unet[18] and SENet - Squeeze and Excitation Network[10]. At the time we are writing this preliminary report, we have not yet tested these approaches to tell if they really work with our model.

3.2 Training procedure

3.2.1 Surrogate Gradient Learning

Because of their binarity and threshold mechanics, SNNs have long been dissuasive against the deep learning community, unable to use the backpropagation algorithm through non-differentiable spiking layers. Instead, SNNs investigations were mostly limited to unsupervised learning with STDP rule [14], or semi-supervised learning using dopamine reward signals such as R-STDP [15]. However, the recent introduction of Surrogate Gradient (SG) learning[16] finally dodges these limitations and allows SNNs to be trained like any other ANN with BP, potentially on supervised learning task[23]. Essentially, SG learning uses a smooth, differentiable surrogate gradient to approximate that of firing threshold.

This simple idea has proven to be an extremely efficient way of training SNNs on a wide range of tasks that once were the privilege of regular ANNs. SG learning has been proven to be robust to many factors, such as the shape of the surrogate derivative, loss functions, input paradigms or datasets[24].

Overall, thanks to this rule, SNNs are bridging the gap in performances with ANNs, at the benefit of high computational and energy efficiency. Also, even if BP is not a biologically plausible learning rule for training, in inference SNNs remain much more faithful to biology than ANNs.

3.2.2 From BP to TBPTT

Back-Propagation Through Time (BPTT) is the ubiquitous method for training Recurrent Neural Networks (RNN). In such models, gradients of the error between the prediction and the label must be not only spatially, but also temporally back-propagated. This is because of the recurrent connections: the state of a unit at time t not only depends on its inputs but also of its previous state at time $t - dt$. To apply this procedure, one must first unroll the

recurrent network to get a feedforward network as in Fig 3.1; it is now trainable with regular back-propagation (BP). SNNs can indeed be seen as a particular case of RNNs, so they are trainable with this method. However, it is not possible to discretize time to arbitrarily small steps. Indeed, the smaller the timestep dt, the larger the unrolled graph, and the more memory is needed for BP. This constitutes a major limitation in the training of RNNs and SNNs, that is partially addressed with a variant method known as Truncated Back-Propagation Through Time (TBPTT).

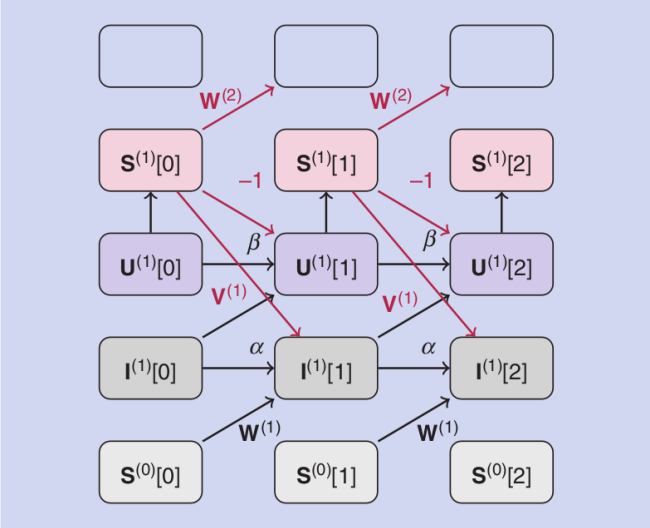

Fig. 3.3 Unrolled computational graph of an SNN in discrete time. Time flows from left to right. Superscripts in brackets denote the layer number. Borrowed from [23].

Let us first consider a network that processes the input sequence one timestep at a time. In TBPTT, the model does k1 forward passes (i.e., sees k1 timesteps), after which a loss is computed by unrolling the model accross k2 timesteps; gradients are then backpropagated,

weights updated accordingly, and the process is repeated.

Algorithm 1: classical TBPTT training procedure

Input: parameters k1, k2;
initialize network states;
while *sequence not finished* **do**
    forward-propagate k1 timesteps;
    calculate loss;
    backward-propagate k2 timesteps;
    update weights;
    detach the graph;
end

As a result, the TBPTT algorithm requires the consideration of two parameters:

- k_1: The number of forward-pass timesteps between updates. Generally, this influences how slow or fast training will be, given how often weight updates are performed.
- k_2: The number of timesteps to which to apply BPTT. Generally, it should be large enough to capture the temporal structure in the problem for the network to learn.

If this training method is conceptually simple, we have been unable to find a working example in PyTorch, nor to implement it with $k_1 < k_2$. The difficulty comes from very architecture of the core of Pytorch. We were very surprised to see that the scientific community has not investigated this particular case of TBPTT which we believe to yield better results, and instead sticked to the case $k_1 = k_2$. Because we believe that a k_2 larger than k_1 would help the network learn finer temporal dependencies in a more continuous fashion, we propose an approximation of this case compatible with PyTorch. The main idea is simply to use the batch dimension to train on shifted versions of the same training sequence (see figure below).

Pytorch codes for this procedure are available on our Github repository. We used this method throughout this project, and made it general enough to encompass the case $k_1 = k_2$. We noticed that convergence was faster using this method, and sometimes lead to better performances. However, we still have to benchmark our algorithm to quantify its benefits over the classical version of TBPTT. This a study that we plan to do in the future of this master's thesis.

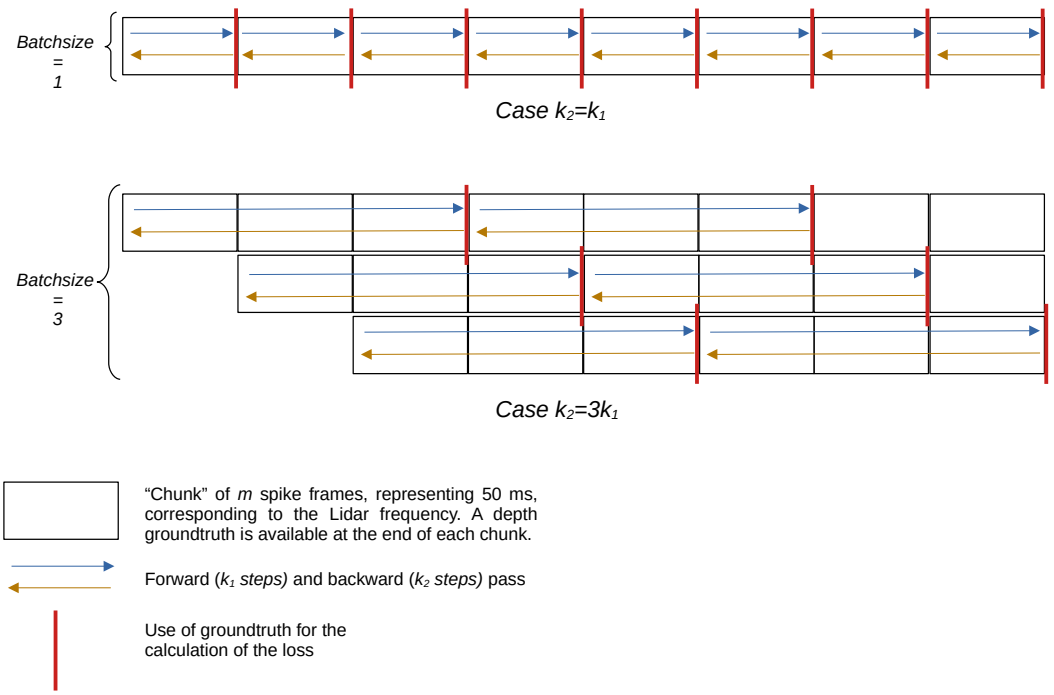

Fig. 3.4 Schematic of our approximated back-propagation through time procedure. Time flows from left to right. The particular case $n = 1$ is nothing else than classical TBPTT. For $n > 1$, we use the batch dimension to train on n shifted versions of the same sequence with a classical TBPTT case of $k_2 = k_1 = n$ chunk. All labels are used with this method, which we believe to be a condition of success.

3.2.3 Loss

Similarly as in [11], we use a combination of a multiscale regression loss with a multiscale regularization loss. Noting $R = \hat{D} - D$ the residual between the groundtruth depth map and prediction at scale s, the first term can be written as:

$$L_{regression} = \frac{1}{n}\sum_s(\sum_u(R(u))^2 - \frac{1}{n^2}(\sum_u R(u))^2) \tag{3.1}$$

where n is the number of valid groundtruth pixels u. With the same notations, the regularization loss is computed with:

$$L_{smooth} = \frac{1}{n}\sum_s\sum_u |\nabla_x R_s(u)| + |\nabla_y R_s(u)| \tag{3.2}$$

According to [11], the minimization of this term encourages smooth depth changes as well as sharp depth discontinuities in the depth map prediction, hence helping the network to represent objects that stand out of the background of the scene (e.g., because they are closer), while respecting its overall topology. Finally, we weight both terms with a factor λ in the total loss:

$$L_{tot} = L_{regression} + \lambda L_{smooth} \tag{3.3}$$

We have yet to find a good empirical value for λ; for the time being we use a value of 0.5. Groundtruth depth maps are downsampled to the intermediary scales with bilinear interpolation.

3.2.4 Network Initialization

In fact, our network does not learn to predict the depth of the scene from one time to another, but rather the depth changes between the two. Thus, in order to prevent it from spending most of its expressive power in initially adapting the potentials of its output neurons to the groundtruth, we initialize these potentials with the first groundtruth depth map of the sequence before training on it. In short, this helps learn the "steady" regime, not the "transient". Other papers such as [7] let the network warm-up with the beginning of the sequence before evaluating losses and backpropagating gradients.

Although this method helps the model behave better in training mode, it poses an ethical problem in evaluation: ideally, it should not use any knowledge of the test set before performing inference on it. However, one could imagine to rather initialize the network with a "standard" depth prior that would be computed from statistics of natural scenes. Several studies have identified common features in natural scenes, such as a depth gradient from the bottom to the top of the field of view -nearer and farther objects being located respectively in the lower and upper half of the scene. Different depth priors could be obtained for indoor and outdoor scenarii, by averaging corresponding MVSEC data for instance, or using principal components.

3.2.5 Temporal Mirroring of Data

In our initial tests, we used the previous network initialization before feeding the network with an input sequence in chronological order. At the end of the sequence, we re-initialized it to its beginning state, fed it with the same sequence from start to finish, re-initialized it, and repeated again. With this method, we noticed that after some epochs, the network could

predict at a certain time objects that appear much later in the sequence. We assume that this phenomenon is due to the discontinuous re-initialization: at the beginning of the next epoch, the latest weight updates were done for late data where specific objects appear.

To address this problem, we propose a new, an easy-to-implement trick for training a network on a long sequence. We first initialize the network like above, and feed it with the training sequence in chronological order from start to finish. From here, the network is not re-initialized, and the data is given in reverse chronological order, from finish to start. This essentially constitutes one epoch. We do not re-initialize the network between two epochs with this procedure. This way, there is no sharp discontinuity in the states of the model and what it has recently learnt. We believe that in addition to potentially resolving the problem mentioned above, this temporal mirroring of training sequences also serves the purpose of an efficient data augmentation technique. Indeed, a ideal model must able to predict depth changes in both "forward" and "backward" ways.

A final and important remark that is particular to DVS data with this trick, is to revert the polarities of the events when feeding the sequence backward in time.

3.2.6 Hyperparameters

We train for 15 epochs on the first 260 groundtruths of *indoor_flying _1* sequence. Our first objective being to make our model overfit and have satsifying performances on this sequence, we also used as our test set. During training, we use ADAM optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and a weight decay of 0.01. We use a multi-step learning rate scheduler with an initial value of 0.0001 and $\gamma = 0.1$. The network is trained for 15 epochs with initial initialization and temporal mirroring for smooth training and data augmentation. We adopt our approximated TBPTT procedure with $nfpdm = 5$ timesteps, $k_1 = 5$ timesteps and $k_2 = 20$ timesteps. PLIF neurons are initialized with a time constant of 10 timesteps.

3.3 Results

3.3.1 Performances

In the final version of this report, we will present here a comparison of the performances of our model against those of competitors. However, we could not report any meaningful quantitative results at the time we are writing these lines, because we do not have a version of our SNN that is good enough for this purpose.

Until now, we have been busy looking for a good formulation of the network that could perform well on a very small subset of the MVSEC dataset. In short, the performances of

our best model (0.68 m in mean depth error) barely surpass that of using the first groundtruth of the training sequence as a constant prediction (0.73 m mean depth error).

It appears that to make the loss decrease, our model predicts a slowly changing depth image, that lacks dynamism as mentioned earlier. We attribute this phenomemon to the discrete nature of SNNs and the layers preceding the pool of I neurons, whose potentials bear the depth predictions. Our absolute priority for the following of the internship is to find a good architecture formulation, that would allow enough expressivity in our model.

Nevertheless, we must say that we have done most work during this first period: we studied the litterature, proposed first trainable versions of our model, found innovative tricks that could help in the success of our approach... Finding the right model is really the bottleneck of our internship, and after this step, we can continue our studies, especially in the field of neuroscience.

3.3.2 Computational and Energy Efficiency

In the long term, we intend to do a complete study of the computational and energy efficiency achieved by our SNN compared to an equivalent ANN, in the fashion of [12]. But at this stage of the internship, it is premature to conduct such a study, for we have not retained a satisfying architecture in terms of performances.

Chapter 4

Conclusion

4.1 Achieved work

To conclude this preliminary report, we would like to say that we have done a quantity of things: train on SNNs and the biology of depth perception, get informed on the state-of-the-art, implement trainable versions of a model as well as a complex training procedure...

4.2 Future Work

However, many things remain to be done, the most important being finding a good model architecture as discussed previously. We have many studies to conduct, listed in a priority order:

1. find a model architecture with good performances (by mid-June hopefully)
2. get an optimized version that can compare to other deep learning approaches (by the end of June)
3. conduct more neuroscience-oriented studies on this model (by mid-August)
4. benchmark our approximated TBPTT procedure on our model and MVSEC dataset if time allows (estimated time: 1 week)

Time flows fast, and we hope we find a good model architecture soon !

References

[1] Bowers, J. S. (2017). Parallel distributed processing theory in the age of deep networks. *Trends in Cognitive Sciences*, 21(12):950–961.

[2] Chauhan, T., Héjja-Brichard, Y., and Cottereau, B. R. (2020). Modelling binocular disparity processing from statistics in natural scenes. *Vision Research*, 176:27–39.

[3] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16.

[4] Fang, W. (2020). Leaky integrate-and-fire spiking neuron with learnable membrane time parameter. *CoRR*, abs/2007.05785.

[5] Fang, W., Chen, Y., Ding, J., Chen, D., Yu, Z., Zhou, H., Tian, Y., and other contributors (2020). Spikingjelly. https://github.com/fangwei123456/spikingjelly.

[6] Fang, W., Yu, Z., Masquelier, T., Chen, Y., Huang, T., and Tian, Y. (2021). Spike-based residual blocks.

[7] Gehrig, M., Shrestha, S. B., Mouritzen, D., and Scaramuzza, D. (2020). Event-based angular velocity regression with spiking networks. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4195–4202.

[8] Godard, C., Aodha, O. M., and Brostow, G. J. (2017). Unsupervised monocular depth estimation with left-right consistency. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6602–6611.

[9] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[10] Hu, J., Shen, L., Albanie, S., Sun, G., and Wu, E. (2020). Squeeze-and-excitation networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(8):2011–2023.

[11] Javier Hidalgo-Carrio, D. G. and Scaramuzza, D. (2020). Learning monocular dense depth from events. *IEEE International Conference on 3D Vision.(3DV)*.

[12] Lee, C., Kosta, A., Zhu, A. Z., Chaney, K., Daniilidis, K., and Roy, K. (2020). Spike-flownet: Event-based optical flow estimation with energy-efficient hybrid neural networks. *CoRR*, abs/2003.06696.

[13] Lee, C., Kosta, A. K., and Roy, K. (2021). Fusion-flownet: Energy-efficient optical flow estimation using sensor fusion and deep fused spiking-analog network architectures. *CoRR*, abs/2103.10592.

[14] Masquelier, T., Guyonneau, R., and Thorpe, S. J. (2008). Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains. *PLOS ONE*, page 1377.

[15] Mozafari, M., Ganjtabesh, M., Nowzari-Dalini, A., Thorpe, S. J., and Masquelier, T. (2019). Bio-inspired digit recognition using reward-modulated spike-timing-dependent plasticity in deep convolutional networks. *Pattern Recognition*, 94:87–95.

[16] Neftci, E. O., Mostafa, H., and Zenke, F. (2019). Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63.

[17] Odena, A., Dumoulin, V., and Olah, C. (2016). Deconvolution and checkerboard artifacts. *Distill*.

[18] Oktay, O., Schlemper, J., Folgoc, L. L., Lee, M. J., Heinrich, M., Misawa, K., Mori, K., McDonagh, S. G., Hammerla, N., Kainz, B., Glocker, B., and Rueckert, D. (2018). Attention u-net: Learning where to look for the pancreas. *ArXiv*, abs/1804.03999.

[19] Ramachandran, V. S. (1988). Perceiving shape from shading. *Scientific American*, 259:76–83.

[20] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597.

[21] Tulyakov, S., Fleuret, F., Kiefel, M., Gehler, P., and Hirsch, M. (2019). Learning an event sequence embedding for dense event-based deep stereo. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1527–1537.

[22] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

[23] Zenke, F., Bohté, S. M., Clopath, C., Comşa, I. M., Göltz, J., Maass, W., Masquelier, T., Naud, R., Neftci, E. O., Petrovici, M. A., Scherr, F., and Goodman, D. F. (2021). Visualizing a joint future of neuroscience and neuromorphic engineering. *Neuron*, 109(4):571–575.

[24] Zenke, F. and Vogels, T. P. (2021). The Remarkable Robustness of Surrogate Gradient Learning for Instilling Complex Function in Spiking Neural Networks. *Neural Computation*, 33(4):899–925.

[25] Zhu, A. Z., Thakur, D., Özaslan, T., Pfrommer, B., Kumar, V., and Daniilidis, K. (2018). The multivehicle stereo event camera dataset: An event camera dataset for 3d perception. *IEEE Robotics and Automation Letters*, 3(3):2032–2039.

[26] Zhu, A. Z., Yuan, L., Chaney, K., and Daniilidis, K. (2018). Ev-flownet: Self-supervised optical flow estimation for event-based cameras. *CoRR*, abs/1802.06898.