ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

**SCUOLA DI INGEGNERIA E ARCHITETTURA**
DIPARTIMENTO DELL'ENERGIA ELETTRICA E DELL'INFORMAZIONE
INGEGNERIA ELETTRONICA E TELECOMUNICAZIONI

Corso di laurea magistrale in

**ELECTRONIC ENGINEERING FOR INTELLIGENT SYSTEMS, BIG-DATA AND INTERNET OF THINGS**

# Spiking Neural Networks for Ultra Low Power Event-Based Optical Flow Estimation

MASTER'S THESIS IN

**ARCHITECTURES FOR ARTIFICIAL INTELLIGENCE**

**Lorenzo Squarzoni**

**Supervisors:**
Dr. Lorenzo Lamberti, llamberti@iis.ee.ethz.ch
Dr. Francesco Barchi, francesco.barchi@unibo.it
Dr. Daniele Palossi, dpalossi@iis.ee.ethz.ch

**Professors:**
Prof. Dr. Francesco Conti, f.conti@unibo.it
Prof. Dr. Andrea Acquaviva, andrea.acquaviva@unibo.it
Prof. Dr. Luca Benini, lbenini@iis.ee.ethz.ch

Sessione V

Anno accademico 2024/2025

# Abstract

Event-based cameras provide an asynchronous, high-temporal-resolution sensing modality that is well suited for low-latency and low-power perception. In parallel, spiking neural networks (SNNs) offer a biologically inspired computational model that naturally aligns with event-driven data and neuromorphic hardware. Together, they represent a promising foundation for energy-efficient motion perception in embedded systems. However, deploying spiking neural networks for dense optical flow estimation on ultra-low-power hardware remains challenging due to the combined demands of accuracy, memory footprint, recurrent state management, and hardware compatibility.

This thesis investigates the feasibility of spiking neural networks for event-based optical flow estimation with a strong focus on deployment-oriented constraints. Starting from a state-of-the-art self-supervised SNN architecture, the work conducts a systematic exploration spanning model architecture, input resolution, temporal windowing, evaluation metrics, and quantization. Particular emphasis is placed on integrating Leaky Integrate-and-Fire (LIF) neuron layers into a software-to-hardware toolchain, enabling realistic simulation of memory usage, computational cost, and throughput on PULP-based ultra-low-power platforms.

A comprehensive trade-off analysis reveals that channel reduction and spatial resolution scaling provide substantial memory and computational savings with limited impact on accuracy, while recurrent spiking layers remain critical for preserving motion direction fidelity. Post-Training Quantization, when combined with careful calibration of LIF neuron states, reduces static memory requirements by approximately 4× with negligible accuracy degradation. Hardware-oriented simulations on the Siracusa PULP platform demonstrate that optimized and quantized models can achieve inference rates ranging from 0.1 Hz for large floating-point configurations up to 150 Hz for compact int8 models, with several configurations fitting entirely within on-chip L2 memory.

Overall, this work demonstrates that spiking neural networks can perform dense event-based optical flow estimation within the stringent constraints of ultra-low-power embedded hardware, provided that architectural design, quantization, and memory hierarchy

are jointly considered. The results bridge the gap between algorithmic SNN research and practical deployment, advancing the applicability of neuromorphic vision systems in real-world scenarios.

# Contents

# List of Figures

# List of Tables

# List of Acronyms

AAE . . . . . Average Angular Error

AE . . . . . Angular Error

AEE . . . . . Average Endpoint Error

ANN . . . . . Artificial Neural Network

BPTT . . . . . BackPropagation Through Time

EE . . . . . Endpoint Error

GPU . . . . . Graphics Processing Unit

HDR . . . . . High Dynamic Range

IWE . . . . . Image of Warped Events

LIF . . . . . Leaky Integrate-and-Fire

MAC . . . . . Multiply-Accumulate

MVSEC . . . . . Multi Vehicle Stereo Event Camera

NCHW . . . . . N. of batches, Channels, Height, Width

NHWC . . . . . N. of batches, Height, Width, Channels

ONNX . . . . . Open Neural Network Exchange

PTQ . . . . . Post-Training Quantization

PULP . . . . . Parallel Ultra-Low-Power

QAT . . . . . Quantization-Aware Training

RGB . . . . . Red Green Blue

SNN . . . . . Spiking Neural Network

Chapter 1

# Introduction

Event-based vision has emerged as a promising paradigm for perception in scenarios where low latency, high temporal resolution, and energy efficiency are critical. In parallel, spiking neural networks have gained increasing attention as a computational model that naturally aligns with event-driven sensing and neuromorphic hardware. This thesis investigates the application of spiking neural networks to the problem of event-based optical flow estimation, with a specific focus on ultra-low-power embedded deployment.

## 1.1. Motivation

Optical flow estimation is a fundamental component of many perception systems, enabling tasks such as motion estimation, navigation, and scene understanding. In robotics and autonomous systems, optical flow is often required to operate under stringent constraints in terms of latency, power consumption, and robustness to challenging lighting conditions.

Event cameras address several limitations of traditional frame-based sensors by asynchronously reporting pixel-level brightness changes with microsecond resolution and high dynamic range. These properties make event-based vision particularly well suited for high-speed and low-power applications. However, the unconventional data format produced by event cameras requires algorithms that differ substantially from those designed for synchronous image frames.

At the same time, spiking neural networks offer a biologically inspired computational model that processes information through sparse, event-driven activations. When deployed on neuromorphic or ultra-low-power hardware, SNNs have the potential to significantly reduce energy consumption compared to conventional artificial neural networks.

The combination of event cameras and spiking neural networks therefore represents a natural and compelling direction for energy-efficient perception.

Despite this strong conceptual alignment, most existing solutions for event-based optical flow either rely on conventional ANN architectures or focus on algorithmic feasibility without considering deployment constraints. Bridging the gap between accurate optical flow estimation and practical ultra-low-power deployment remains an open challenge.

## 1.2. Problem Statement

The problem addressed in this thesis is the estimation of dense optical flow from event-based camera data under strict hardware constraints. Specifically, the goal is to design and evaluate a spiking neural network capable of processing event streams and producing accurate optical flow estimates while respecting the memory, computational, and data-layout limitations of ultra-low-power embedded platforms.

This problem involves several interconnected challenges. First, dense optical flow estimation is a regression task that requires precise spatial and temporal reasoning, which is inherently difficult for spiking neural networks. Second, event-based data are sparse and asynchronous, complicating both input representation and evaluation. Third, the internal state and recurrence present in spiking neuron models introduce significant memory overhead, which must be carefully managed to enable deployment on constrained hardware.

Finally, translating a trained SNN model from a high-level software framework to a hardware-executable representation requires addressing compatibility issues related to model export, operator support, quantization, and memory hierarchy.

## 1.3. Research Objectives

The primary objective of this thesis is to conduct a comprehensive, deployment-oriented exploration of spiking neural networks for event-based optical flow estimation. Rather than focusing exclusively on maximizing accuracy, the work aims to understand and quantify the trade-offs between performance, memory usage, and computational complexity.

More specifically, the objectives of this work are to:

- Reproduce and analyze a state-of-the-art spiking neural network architecture for event-based optical flow estimation.
- Investigate architectural modifications, such as channel reduction, resolution scaling and recurrence removal, to improve suitability for embedded deployment.

- Study and apply appropriate evaluation metrics for optical flow, with particular attention to their behavior under varying temporal and spatial resolutions.
- Develop and validate quantization strategies for recurrent spiking neural networks, including the handling of neuron states.
- Integrate spiking neuron models into a software-to-hardware toolchain and simulate their execution on ultra-low-power platforms.

## 1.4. Contributions

The main contributions of this thesis can be summarized as follows:

1. **End-to-end analysis of spiking optical flow models**
   A detailed exploration of spiking neural network architectures for event-based optical flow estimation, starting from a state-of-the-art model and extending it with deployment-oriented considerations.
2. **Architectural trade-off study**
   A systematic evaluation of how model size, spatial resolution, recurrence, and depth affect accuracy, memory footprint, and computational cost.
3. **Metric-driven evaluation framework**
   An analysis of optical flow evaluation metrics and their sensitivity to temporal windowing and spatial resolution, enabling fair and interpretable comparisons across configurations.
4. **Quantization of recurrent SNNs**
   The development and validation of post-training quantization strategies for spiking neural networks, including correct handling of LIF neuron states and parameters.
5. **Software-to-hardware integration**
   The integration of LIF neuron layers into ONNX[1] and Deeploy[2], enabling realistic simulation and profiling of spiking models on PULP-based platforms [2].

Together, these contributions advance the practical applicability of spiking neural networks for low-power event-based vision.

[1] https://onnx.ai
[2] https://github.com/pulp-platform/Deeploy

## 1.5. Thesis Structure

The remainder of this thesis is organized as follows.

Chapter 2 reviews related work in event-based optical flow estimation, spiking neural networks for vision, hybrid ANN–SNN architectures, and neuromorphic hardware deployment.
Chapter 3 provides the theoretical background on event cameras, optical flow, spiking neuron models, self-supervised learning, evaluation metrics, and hardware constraints.
Chapter 4 describes the proposed methodology, including model architecture, training and evaluation pipelines, architectural exploration, quantization strategies, and hardware-oriented simulation.
Chapter 5 presents the experimental results, analyzing the trade-offs between accuracy, memory usage, and computational efficiency.
Finally, Chapter 6 discusses the results, highlights limitations, compares the proposed approach with the state of the art, and outlines directions for future work.

Chapter 2

# Related Work

This chapter reviews prior work relevant to event-based optical flow estimation, spiking neural networks for vision, hybrid ANN–SNN approaches, and neuromorphic hardware implementations. The objective is to contextualize the proposed work within the state of the art and to highlight the limitations that motivate the contributions of this thesis.

## 2.1. Event-Based Optical Flow Estimation

Optical flow estimation from event cameras has been an active research topic since the introduction of neuromorphic vision sensors. Early approaches relied on handcrafted algorithms exploiting the spatiotemporal structure of events, often based on local plane fitting in the space–time domain or motion compensation techniques. While computationally efficient, these methods typically struggle in complex scenes and lack robustness to noise [3].

With the rise of deep learning, learning-based approaches have become predominant. One of the first convolutional neural network architectures for event-based optical flow estimation introduced an encoder–decoder structure trained in a self-supervised manner using photometric consistency between reconstructed frames [4]. This work was later extended by replacing photometric losses with contrast maximization losses computed directly from events, enabling training without relying on frame-based sensors.

Subsequent works refined these ideas by improving the loss formulation, introducing better event representations, and optimizing the architectures for efficiency. FireNet, for example, was proposed as a lightweight alternative to encoder–decoder models, trading some accuracy for a significant reduction in parameter count and computational complexity [4]. These ANN-based methods achieve high accuracy but rely on dense input

representations and frame-like processing, which limits their suitability for ultra-low-power and fully asynchronous pipelines.

## 2.2. Spiking Neural Networks for Vision

Spiking Neural Networks naturally align with the asynchronous and sparse output of event cameras, making them an attractive candidate for event-based vision tasks. Early SNN-based approaches primarily targeted classification and simple motion detection problems, often using unsupervised learning rules such as spike-timing-dependent plasticity [3, 5].

Applying SNNs to dense regression tasks such as optical flow estimation is considerably more challenging. One of the first attempts involved unsupervised Hebbian learning combined with synaptic delays to induce motion selectivity. While biologically inspired, this approach was limited in scalability and generalization, as it depended heavily on the statistics of the training data [6].

More recent work demonstrated that deep SNNs can be trained using surrogate gradient methods to perform event-based optical flow estimation in a self-supervised fashion. These models adopt architectures inspired by ANN counterparts while replacing activation functions with spiking neuron models, such as the Leaky Integrate-and-Fire neuron. Despite achieving accuracy comparable to ANN-based approaches, these works primarily focus on algorithmic feasibility and training stability, with limited consideration of hardware deployment constraints [4, 7, 8].

## 2.3. Hybrid ANN-SNN Architectures

To mitigate the training difficulty of fully spiking networks, several works have proposed hybrid ANN–SNN architectures. In these models, spiking neurons are typically used in early layers to process event-based input, while conventional ANN layers handle higher-level representations and output decoding [9, 10, 11].

Hybrid approaches have been shown to improve training stability and accuracy, and they often simplify deployment on conventional hardware. Some works also explore sensor fusion, combining event-based and frame-based inputs to leverage complementary information. While these methods achieve strong performance, they compromise the goal of a fully neuromorphic pipeline and still rely on dense, synchronous processing stages [9].

From a hardware perspective, hybrid architectures reduce the potential energy savings offered by SNNs, as ANN components dominate memory accesses and arithmetic operations. As a result, they are less suitable for ultra-low-power platforms where energy efficiency and sparse computation are primary concerns [10].

## 2.4. Neuromorphic Hardware and Low-Power Inference

Several studies have investigated the execution of neural networks on neuromorphic or low-power hardware platforms. Surveys on neuromorphic computing provide extensive overviews of spiking neuron models, learning mechanisms, and hardware architectures, highlighting the potential efficiency gains of event-driven computation [5].

A small number of works have reported real hardware implementations of event-based optical flow systems, often using specialized neuromorphic chips or hybrid setups combining neuromorphic processors and GPUs. These implementations demonstrate significant energy savings but typically rely on hand-crafted algorithms or fixed-function pipelines rather than learned models [12].

More recent efforts have focused on hardware-aware neural network design, including model compression, pruning, and quantization. While quantization of ANNs is well established, applying similar techniques to recurrent SNNs remains challenging due to the presence of internal neuron states and non-linear dynamics. As a result, many SNN-based works stop at software-level evaluation or use simplified neuron models that are easier to deploy [13, 14].

## 2.5. Summary and Positioning

Despite significant progress in event-based optical flow estimation, several gaps remain. ANN-based methods achieve high accuracy but are not well suited for ultra-low-power deployment. Hybrid ANN–SNN approaches reduce training complexity but sacrifice the advantages of fully spiking computation. Pure SNN solutions demonstrate feasibility but often neglect hardware constraints, quantization, and memory management.

This thesis addresses these limitations by focusing on a deployment-oriented exploration of spiking neural networks for event-based optical flow estimation. Starting from a state-of-the-art SNN model, the work systematically investigates architectural trade-offs, error metrics, quantization strategies, and hardware-aware execution. In particular, it emphasizes the integration of LIF neuron layers into a software–hardware toolchain targeting PULP-based platforms, enabling realistic simulation of memory usage, latency, and throughput [4].

By bridging the gap between algorithmic SNN design and hardware-oriented deployment, this work contributes toward making event-based optical flow estimation feasible on ultra-low-power embedded systems.

Chapter 3

# Background

This chapter introduces the theoretical and technical foundations required to understand the problem addressed in this thesis and the proposed solution. It covers event-based vision, optical flow estimation, spiking neural networks, self-supervised learning for event-based optical flow, evaluation metrics, and hardware constraints relevant to ultra-low-power deployment.

## 3.1. Event Cameras

Event cameras, also known as neuromorphic or dynamic vision sensors, differ fundamentally from conventional frame-based cameras. Instead of capturing full image frames at fixed time intervals, event cameras asynchronously record per-pixel brightness changes. Each pixel independently triggers an event whenever the logarithmic intensity change exceeds a predefined threshold [3].

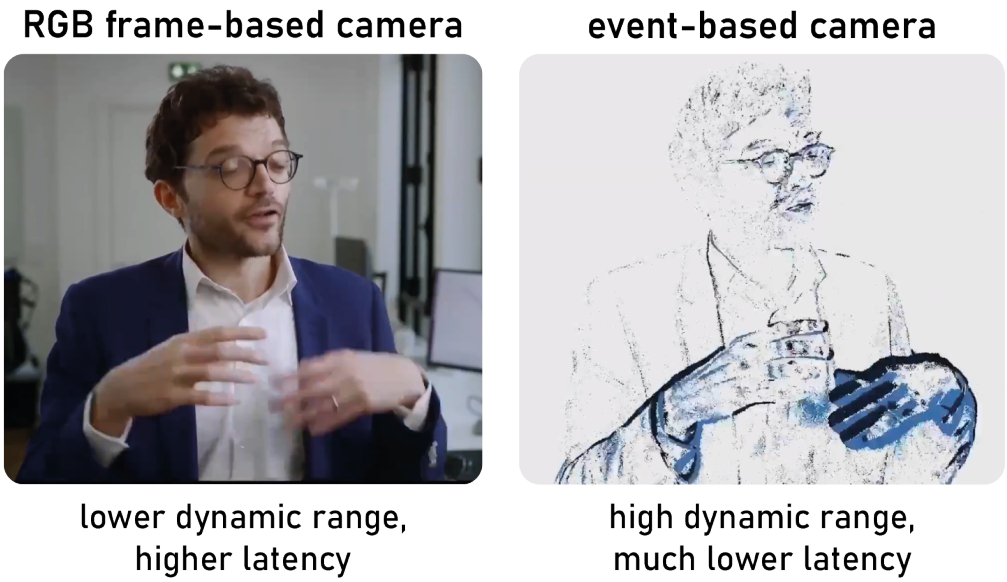


Figure 3.1.: Standard RGB camera vs. event-based camera: each pixel independently emits an event whenever the local logarithmic brightness change exceeds a threshold, usually around the edges, producing an asynchronous stream of events

An event is typically represented as a tuple:

$$e_i = (p_i, t_i, x_i, y_i) \tag{3.1}$$

where $(x_i, y_i)$ denotes the pixel coordinates, $t_i$ the timestamp with microsecond resolution and $p_i \in +1, -1$ the polarity of the brightness change.

This sensing paradigm offers several advantages:

- **Low latency**, as events are emitted immediately after intensity changes
- **High temporal resolution**, often in the order of microseconds
- **Low power consumption**, due to sparse activity
- **High dynamic range**, enabling operation in challenging lighting conditions

However, event cameras do not directly provide absolute intensity information, and their output is inherently sparse and asynchronous. As a consequence, traditional computer vision algorithms designed for dense, synchronous frames cannot be directly applied.

## 3.2. Optical Flow

Optical flow describes the apparent motion of brightness patterns between consecutive observations. Formally, it is defined as a dense vector field:

$$\mathbf{u}(x, y) = (u(x, y), v(x, y)) \tag{3.2}$$

where $u$ and $v$ represent the horizontal and vertical components of motion at each pixel.


Figure 3.2.: Optical flow visualization: gradient-based and vector-based approaches

In frame-based vision, optical flow estimation typically relies on brightness constancy assumptions and spatial smoothness priors. Event-based optical flow differs significantly due to the absence of frames and the asynchronous nature of the data. Instead of tracking intensity patterns over time, event-based methods exploit the spatiotemporal alignment of events generated by moving edges [3].

Estimating optical flow from event cameras is particularly attractive for high-speed and low-power applications, such as robotics and autonomous navigation, but it introduces additional challenges related to data representation, temporal integration, and evaluation.

## 3.3. Spiking Neural Networks

Spiking Neural Networks (SNNs) are inspired by biological neural systems and process information using discrete events, or spikes. Unlike Artificial Neural Networks (ANNs), where neurons communicate using continuous-valued activations, SNNs transmit information through sparse spike trains over time [5].

### 3.3.1. Leaky Integrate-and-Fire Neuron Model

The Leaky Integrate-and-Fire (LIF) neuron is one of the most widely used spiking neuron models due to its simplicity and biological plausibility. Each neuron maintains an internal

state, the membrane potential $V$, which evolves over time according to incoming input currents and a leakage term [5].

In discrete time, the membrane potential update can be expressed as:

$$V_t = V_{t-1} \cdot leak \cdot (1 - Z_{t-1}) + I_t \tag{3.3}$$

where:

- $I_t$ is the input current
- $leak$ controls the decay of the membrane potential
- $Z_{t-1}$ is the spike output at the previous time step

When $V_t$ exceeds a threshold value, the neuron emits a spike and its membrane potential is reset.

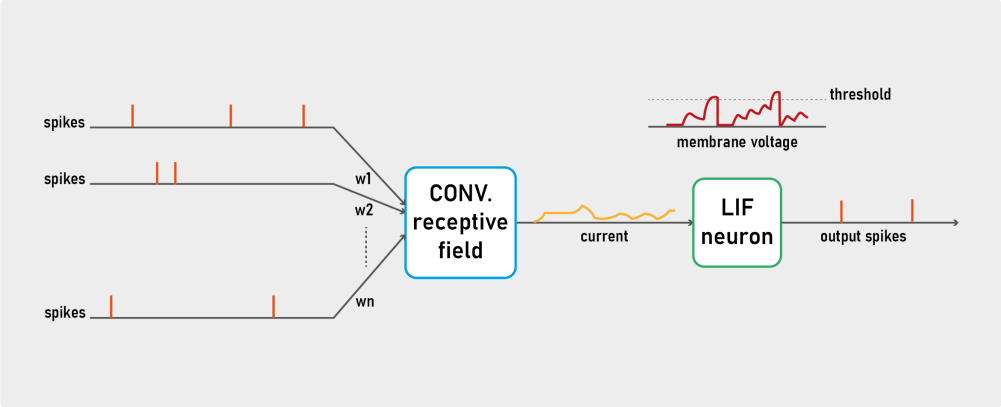


Figure 3.3.: LIF neuron dynamics: example of the state update and reset inside a LIF neuron, with the convolution layer responsible of generating the LIF input current from the incoming spikes (like in our specific model)

In this work, the LIF neuron parameters, such as leak and threshold, are learnable and maintained independently for each output channel of a layer. The neuron state must be preserved across inference steps, introducing memory and state-management requirements that strongly influence hardware deployment.

### 3.3.2. Training Spiking Neural Networks

Training SNNs is challenging due to the non-differentiability of the spike generation function. To enable gradient-based optimization, surrogate gradient methods are commonly

employed. These approaches approximate the derivative of the spiking function during backpropagation, allowing the use of standard optimization algorithms [4].

Despite these techniques, training deep and recurrent SNNs remains more complex than training conventional ANNs, particularly for regression tasks such as optical flow estimation.

## 3.4. Self-Supervised Learning for Optical Flow

Obtaining ground-truth optical flow for event cameras is difficult and often requires additional sensors or synthetic data. For this reason, self-supervised learning approaches have become popular [15].

A widely adopted method is contrast maximization via motion compensation [16]. The core idea is that, given a correct optical flow estimate, events generated by a moving edge can be temporally warped to a common reference time, resulting in a sharp accumulation of events [4].

Given an estimated optical flow field $\mathbf{u}(x, y)$, each event is warped according to:

$$x_i' = x_i + (t_{ref} - t_i) \cdot \mathbf{u}(x_i) \tag{3.4}$$

The warped events are accumulated into an Image of Warped Events (IWE). A sharp IWE indicates accurate motion estimation, while blurred results correspond to incorrect flow.


Figure 3.4.: Motion compensation and IWE generation, with examples of good and bad estimations [1]

The training objective minimizes a contrast-based loss computed on the IWE, often combined with a spatial smoothness regularization term. This approach allows learning optical flow directly from event data, without requiring explicit ground truth.

## 3.5. Optical Flow Evaluation Metrics

Evaluating optical flow estimation performance is non-trivial, especially when comparing models operating at different temporal resolutions, spatial resolutions, or input representations.

### 3.5.1. Average Endpoint Error

The Average Endpoint Error (AEE) measures the Euclidean distance between estimated and ground-truth flow vectors; the result is computed averaging the per-pixel Endpoint Error (EE) by the number of valid pixels in the frame [17]:

$$EE = \sqrt{(u - u_{GT})^2 + (v - v_{GT})^2} \tag{3.5}$$

While widely used, $AEE$ does not account for the magnitude of the motion and can yield identical error values for significantly different flow scenarios.

### 3.5.2. Average Angular Error

The Average Angular Error (AAE) evaluates the angle between estimated and ground-truth vectors. This metric emphasizes directional accuracy but is highly sensitive to small motion magnitudes and near-zero vectors [17].

$$AE = cos^{-1}[\frac{(u \cdot u_{GT}) + (v \cdot v_{GT})}{\sqrt{u^2 + v^2} \cdot \sqrt{u_{GT}^2 + v_{GT}^2}}] \tag{3.6}$$

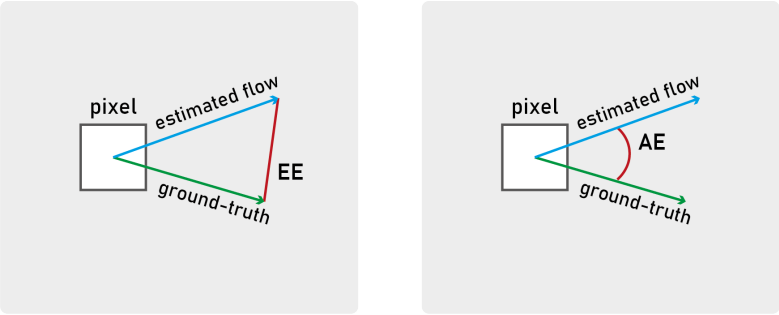


Figure 3.5.: Error metrics: Endpoint Error and Angular Error

Using multiple complementary metrics provides a more comprehensive evaluation of optical flow performance and is particularly important when assessing hardware-oriented trade-offs.

## 3.6. Hardware Constraints for SNN Deployment

Deploying SNNs on ultra-low-power hardware introduces constraints that significantly influence model design [5].

Key considerations include:

- **Memory hierarchy**, typically composed of multiple on-chip memory levels with limited capacity
- **State persistence**, as neuron states must be stored between inference steps
- **Data layout**, such as channel-first (NCHW) versus channel-last (NHWC) representations
- **Tiling and parallelism**, required to execute large tensors within limited memory
- **Quantization**, to reduce memory footprint and computational cost

In particular, recurrent LIF-based architectures require careful management of neuron states and intermediate activations. These constraints motivate hardware-aware architectural choices and optimization strategies, which are central to the methodology presented in this thesis.

Chapter 4

# Methodology

This chapter describes the methodology adopted in this thesis to explore spiking neural networks for ultra-low-power event-based optical flow estimation. Starting from a state-of-the-art reference model, the chapter details the architectural choices, training and evaluation procedures, model exploration strategies, quantization approaches, and the software-to-hardware translation pipeline used to assess deployability on PULP-based platforms.

## 4.1. Reference Architecture

As a starting point, this work builds upon the self-supervised event-based optical flow framework proposed by Paredes-Vallés et al. The framework provides both ANN and SNN variants of well-known architectures, enabling a direct comparison between conventional and spiking approaches under a unified training and evaluation pipeline [4].

Three main architectures are considered:

- **EV-FlowNet**, a deep encoder-decoder network with skip connections
- **FireNet**, a lightweight sequential convolutional architecture
- **FireFlowNet**, a further simplified feed-forward variant, where there is no explicit recurrence

EV-FlowNet achieves strong performance but contains on the order of tens of millions of parameters, making it unsuitable for embedded deployment. FireFlowNet is significantly lighter but suffers from a noticeable degradation in accuracy. FireNet represents a compromise between these two extremes, balancing model size and performance.

Given the hardware-oriented objectives of this thesis, FireNet was selected as the reference architecture for further investigation.

## 4.2. Spiking FireNet Architecture

### 4.2.1. Layer Composition

The spiking version of FireNet replaces conventional activation functions with Leaky Integrate-and-Fire (LIF) neurons [4]. Each convolutional layer is composed of:

1. A convolution producing an input current tensor
2. A LIF neuron layer maintaining internal state
3. A non-linear activation limiting the output range

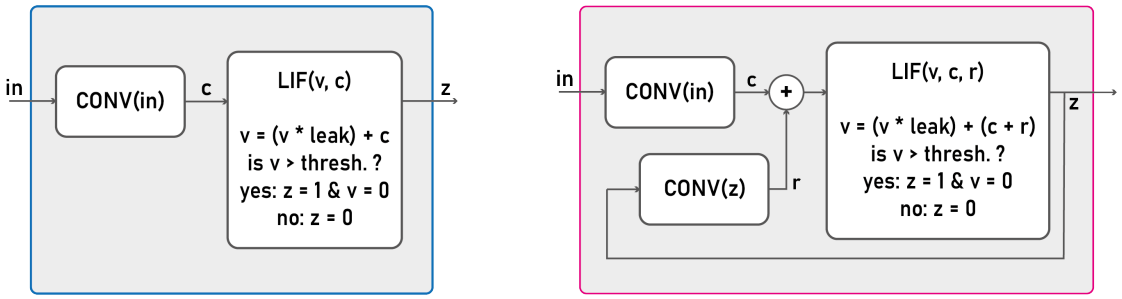


Figure 4.1.: Convolution + LIF blocks schematics: non-recurrent and recurrent variants (c: current, v: membrane voltage, z: output spike, r: recurrent current)

The network is composed of:

- Standard convolutional LIF layers
- Recurrent convolutional LIF layers
- A final simple convolutional output layer, followed by an hyperbolic tangent activation

Recurrent layers introduce temporal integration by feeding back the spike output of the previous timestep through an additional convolution applied to the spike tensor; the output of this convolution is summed to the input current entering the LIF neurons.

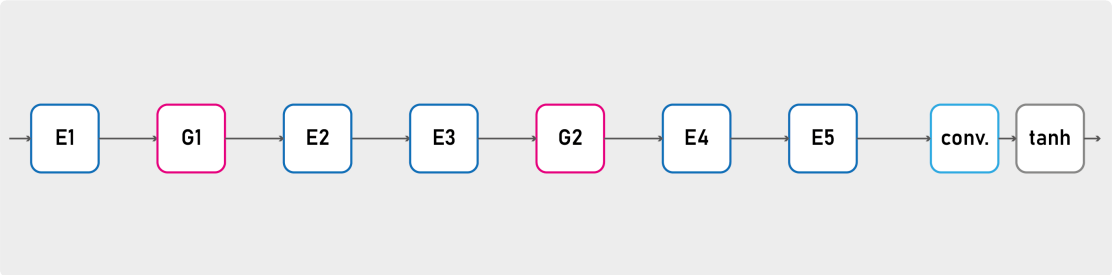


Figure 4.2.: Spiking FireNet baseline model schematic: in blue the non-recurrent convolutional LIF layers, in pink the recurrent ones. The input is the frame of counted events and the output the estimated dense flow.

### 4.2.2. Neuron Parameters and State

Each LIF layer maintains:

- Trainable **leak** parameters
- Trainable **threshold** parameters
- A membrane voltage state tensor
- A spike output tensor

All parameters are maintained per output channel. During inference, the neuron states must persist across consecutive event frames, introducing additional memory requirements compared to stateless ANN layers.

## 4.3. Input Representation and Encoding

The input to the network consists of event frames generated by accumulating events within a temporal or event-count window [4]. Each frame has shape $[1, 2, H, W]$, where the two channels correspond to positive and negative event counts.

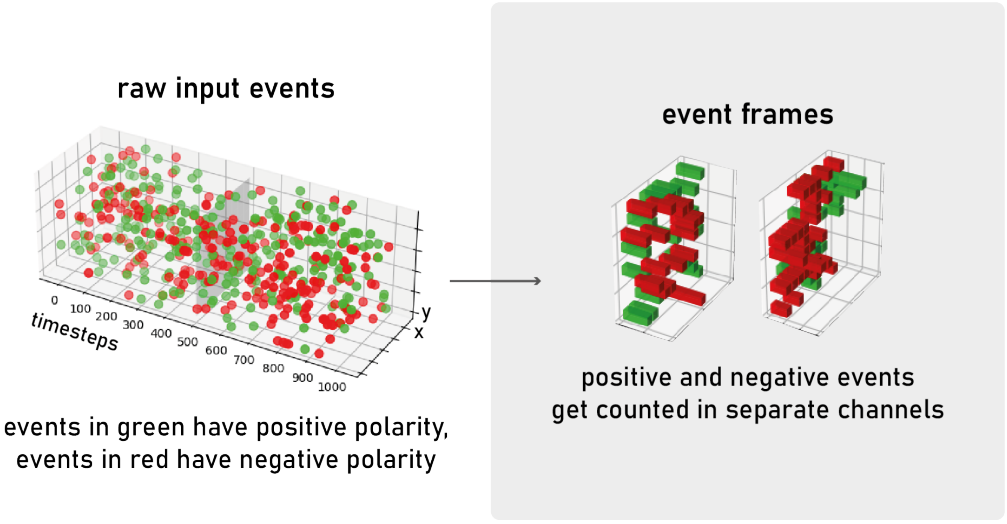


Figure 4.3.: Event-to-frame encoding: fixed a window of accumulation (here 500ms), one frame is generated (here shown as an histogram)

Different encoding modes are supported:

- **Event-count-based encoding**, where frames are generated after accumulating a fixed number of events, usually used at training time
- **Time-based encoding**, where frames represent events within a fixed temporal window
- **Ground-truth-aligned encoding**, where the temporal window matches the sampling rate of ground-truth optical flow, used at evaluation time

Unless otherwise specified, event-count-based encoding is used, as it better preserves the asynchronous nature of the data.

## 4.4. Training Pipeline

### 4.4.1. Self-Supervised Loss Function

Training is performed in a self-supervised manner using a contrast maximization loss via motion compensation [4]. Given an estimated optical flow field, input events are temporally warped to a reference time, generating an Image of Warped Events (IWE).

The loss minimizes the temporal variance of the IWE, encouraging sharp alignment of events corresponding to the same moving edges. A smoothness regularization term is added to enforce spatial coherence in regions with sufficient event density.

The total loss is defined as:

$$\mathcal{L} = \mathcal{L}_{contrast} + \lambda \mathcal{L}_{smooth} \tag{4.1}$$

where $\lambda$ controls the contribution of the regularization term and $\mathcal{L}_{contrast}$ is the sum of a forward and backward warping contributions.

### 4.4.2. Optimization Strategy

At each training iteration:

- A single event frame is fed into the network
- Neuron states are updated during forward inference
- Gradients are computed without backpropagating through time (BPTT) [4]
- Network parameters are updated immediately

This approach simplifies training and limits memory usage but does not exploit long-term temporal dependencies across frames [8].

### 4.4.3. Datasets

Two different datasets are used for training and evaluation, respectively, reflecting the distinct requirements of self-supervised learning and quantitative performance assessment.

#### Training Dataset

For training, the **UZH FPV Drone Dataset**[1] is employed. This dataset provides raw event streams captured from a DAVIS sensor mounted on a quadrotor platform [18], without optical flow ground truth. As the training procedure is fully self-supervised, ground truth is not required. Due to hardware-related constraints and to ensure consistency with later deployment-oriented experiments, the event streams are spatially cropped to a resolution of **128×128 pixels** before being converted into event frames.

[1] https://fpv.ifi.uzh.ch/

### Evaluation Dataset

For evaluation, the **MVSEC (Multi Vehicle Stereo Event Camera) dataset**[2] is used, as it provides ground-truth optical flow obtained through sensor fusion and motion capture systems [19]. From this dataset, four sequences are selected, as they are the most relevant to the target application domain in terms of motion dynamics and scene structure. The spatial resolution of MVSEC is **256×256 pixels**, after cropping it from the original 346x260 size. During evaluation, the data are progressively downsampled to 128×128, 64×64, and 32×32 pixels to analyze the impact of spatial resolution on performance, memory usage, and computational cost.

This separation between training and evaluation datasets ensures that the reported results reflect the generalization capability of the models and avoids overfitting to a single data distribution.

## 4.5. Evaluation Protocol

Model evaluation is performed using multiple complementary metrics [17]:

- **Average Endpoint Error (AEE)**
- **Average Angular Error (AAE)**

To ensure fair comparison across different resolutions and temporal windows, both input frames and ground-truth optical flow are downsampled using average pooling when required.

### Temporal Window Configurations

Evaluation is carried out under two different temporal configurations, defined by the parameter controlling the temporal accumulation window of events [4]:

- **gtflow_dt1**: The temporal window is aligned with the sampling frequency of the RGB camera of the DAVIS sensor. This corresponds to the standard evaluation setting used in the MVSEC dataset and results in relatively small optical flow displacements between consecutive frames.
- **gtflow_dt4**: The temporal window is set to four times the RGB camera sampling period. This leads to larger motion displacements in the ground-truth optical flow and represents a more challenging evaluation scenario, particularly for low-resolution inputs.

[2] https://daniilidis-group.github.io/mvsec

By evaluating models under both configurations, the robustness of the proposed architectures to increasing motion magnitude and temporal integration is assessed.

Visual inspection is supported through color-coded flow maps and vector-field representations, where vector direction and magnitude reflect the estimated motion wrt the ground-truth.

## 4.6. Architecture Exploration

To assess the suitability of the model for embedded deployment, several architectural modifications are explored.

### 4.6.1. Channel Reduction

The number of channels in all convolutional layers, defined as hidden channels and equal for all internal layers of the model, is progressively reduced. This directly impacts:

- Memory footprint
- Number of multiply-accumulate operations (MACs)
- Accuracy

### 4.6.2. Resolution Scaling

The spatial resolution of input frames is reduced to evaluate the trade-off between information loss and computational efficiency. Different strategies for downsampling are evaluated to preserve error coherence.

### 4.6.3. Layer Pruning and Recurrence Analysis

The impact of:

- Removing convolutional layers
- Removing recurrent connections

is evaluated to determine the role of temporal integration and network depth in optical flow accuracy

## 4.7. Quantization Strategy

Given the memory constraints of the target hardware, quantization is a critical component of the methodology [13, 14].

### 4.7.1. Post-Training Quantization

Post-Training Quantization (PTQ) is applied to [14]:

- Convolutional weights and activations
- LIF neuron parameters and states

Special care is taken to correctly calibrate the dynamic range of membrane voltages, which can reach large negative values during inference, probably due to the lack of normalization in the models.

### 4.7.2. Quantization-Aware Training

Quantization-Aware Training (QAT) is explored to reduce accuracy degradation linked to quantization [13]. Several configurations are evaluated, including:

- Mixed precision (quantized convolutions, floating-point LIF layers)
- Fully quantized models

Due to training instability, vanishing gradient and gradient saturation, PTQ is ultimately favored for most configurations.

### 4.7.3. Real Quantized Graph Generation with DeepQuant

While Post-Training Quantization and Quantization-Aware Training allow the evaluation of quantized models within the training framework, hardware-oriented simulation requires the generation of fully quantized computational graphs. For this purpose, the DeepQuant[3] library is employed.

DeepQuant is a tool developed within the PULP research group that enables the transformation of floating-point ONNX graphs into real quantized representations, suitable for execution on embedded platforms. Unlike fake quantization approaches, DeepQuant explicitly inserts quantization, dequantization, and requantization operators into the computational graph.

In the generated graphs:

[3] https://github.com/pulp-platform/DeepQuant

- An initial **Quantize** block converts input tensors from floating-point to int8
- Intermediate **Requantize** blocks are inserted when consecutive layers operate in different quantization domains
- A final **Dequantize** block converts the output back to floating-point

This explicit representation allows accurate modeling of memory usage, data movement, and computational overhead introduced by quantization, which is critical for hardware-oriented profiling.

Since DeepQuant is still under active development, several adaptations were required to support the custom LIF neuron operators introduced in this work.

The resulting quantized ONNX graphs serve as the input to the Deeploy framework and enable realistic simulation of memory allocation, latency, and throughput on PULP-based platforms.

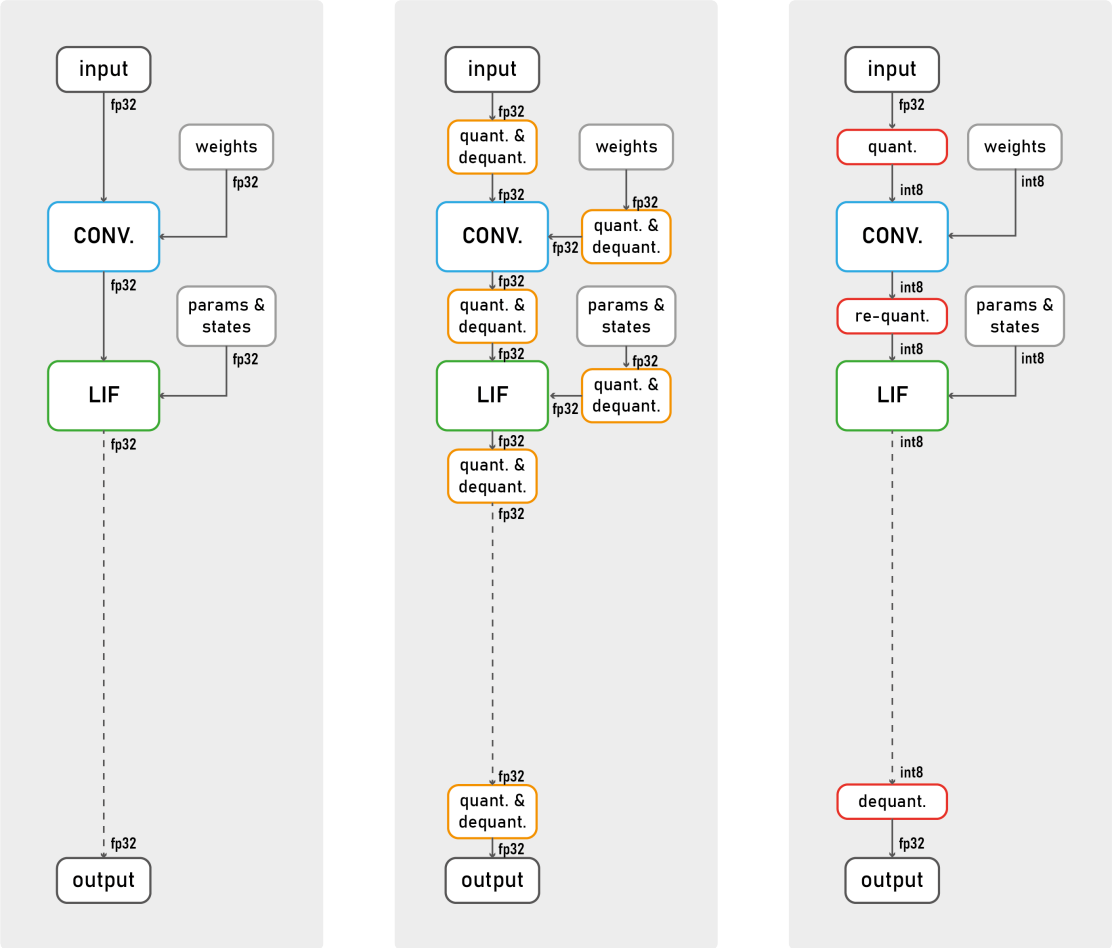


Figure 4.4.: ONNX graph generation: fp32 from Pytorch with custom LIF kernel, fake-quantized int8 from Pytorch and really-quantized int8 from DeepQuant

## 4.8. Software-to-Hardware Translation

### 4.8.1. ONNX Graph Generation

To enable hardware simulation, trained models are exported to ONNX format. Custom operator blocks are introduced to represent LIF neuron layers as single computational units, simplifying graph analysis and conversion.

### 4.8.2. Deeploy Integration

The Deeploy framework is used to translate ONNX graphs into C code suitable for execution on PULP-based platforms. Custom kernels are implemented for:

- LIF neuron layers
- Non-linear activations not natively supported

Both generic and PULP-optimized execution backends are considered.

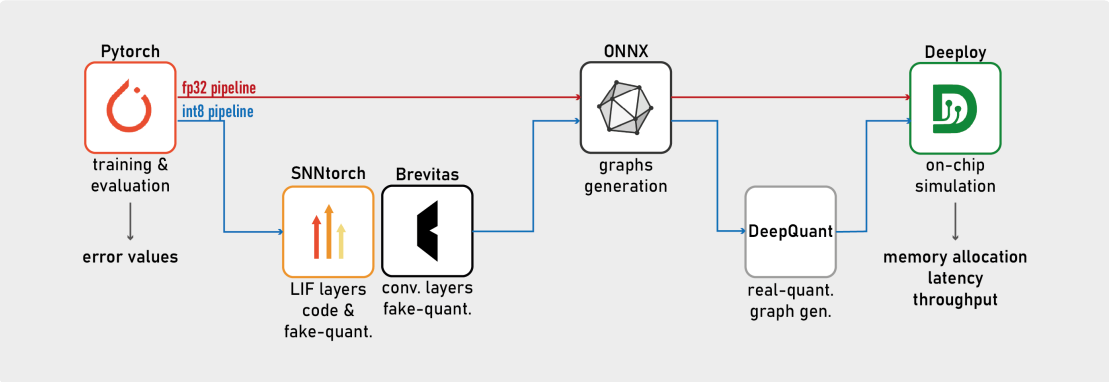


Figure 4.5.: Toolchain overview: from Pytorch to Deeploy

## 4.9. Hardware-Oriented Simulation on PULP

Execution is simulated on PULP platforms using cycle-accurate and memory-aware models. Key aspects include:

- Multi-level memory hierarchy (L1, L2, L3)
- Tiling strategies to fit tensors within on-chip memory
- Tensor layout conversion (NCHW to NHWC)

Performance is evaluated in terms of:

- Static and dynamic memory allocation
- Number of operations (approximated to MACs)
- Achievable inference throughput

Chapter 5

# Results

This chapter presents the experimental results obtained through the methodology described in Chapter 4. The evaluation focuses on the trade-offs between accuracy, memory footprint, and computational complexity, with the goal of assessing the feasibility of deploying spiking neural networks for event-based optical flow estimation on ultra-low-power hardware.

Results are organized to progressively analyze the impact of architectural choices, input resolution, temporal window configuration, and quantization on both performance and hardware-oriented metrics.

## 5.1. Baseline Model Performance

As a reference point, the baseline LIF FireNet architecture is evaluated using its original configuration, before any architectural simplifications or quantization are applied. The only change in the architecture has been the introduction of the Leaky LIF layers from SNN Torch[1] library, which are mathematically equal to the original implementation.

The model is trained on the UZH FPV Drone dataset using self-supervised learning and evaluated on four selected sequences from the MVSEC dataset. Evaluation is performed at our native MVSEC resolution (256×256 pixels) and under both temporal window configurations ($gtflow\_dt1$ and $gtflow\_dt4$).

[1] https://github.com/jeshraghian/snntorch

| AEE ↓ [pixel] | | AAE ↓ [rad] | |
|---|---|---|---|
| dt = 1 | dt = 4 | dt = 1 | dt = 4 |
| 1.17 | 4.23 | 0.42 | 0.40 |

Table 5.1.: Baseline model accuracy (LIF FireNet) for both temporal settings, averaged over 4 MVSEC sequences.

The results confirm that the spiking FireNet architecture is able to achieve optical flow estimation accuracy comparable to previously reported values in the literature, validating the correctness of the training and evaluation pipeline. As expected, performance in terms of Average Endpoint Error degrades when increasing the temporal window ($gtflow_{dt4}$), due to the larger motion magnitudes present in the ground truth. Conversely, the Average Angular Error slightly decreases for longer temporal windows, as larger displacement vectors are less affected by directional ambiguity, in agreement with the metric behavior discussed in Chapter 3.

The results are aligned with those reported in the source paper [4].

## 5.2. Architectural Trade-Offs

To evaluate the suitability of the model for embedded deployment, several architectural modifications are explored and compared against the baseline configuration.

### 5.2.1. Effect of Channel Reduction

Reducing the number of hidden channels has a direct impact on memory usage and computational cost. Models are evaluated with progressively fewer channels (32, 16, 8 and 4) while keeping the overall architecture unchanged.

| N. channels | N. ops [M MAC] | Mem. [MB] | AEE ↓ [pixel] | AAE ↓ [rad - degree] |
|---|---|---|---|---|
| 32 | 7350 | 76.7 | 2.70 | 0.41 - 24 |
| 16 | 1863 | 38.8 | 2.91 | 0.51 - 29 |
| 8 | 478 | 19.9 | 3.21 | 0.72 - 41 |
| 4 | 126 | 10.4 | 3.44 | 1.05 - 60 |

Table 5.2.: Accuracy, Operations and Static Memory Allocation vs. Number of hidden channels @ 256x256 res., averaged over 4 MVSEC seq. and 2 temporal resolutions

Accuracy metrics (AEE and AAE) are obtained from floating-point inference in PyTorch, while the number of operations, static memory allocation, and performance-related metrics are derived from hardware-oriented simulation using Deeploy.

Reducing the number of hidden channels leads to a predictable reduction in computational cost and memory footprint, with the number of operations scaling approximately quadratically and the static memory allocation scaling linearly with the channel count.

From an accuracy perspective, both Average Endpoint Error and Average Angular Error increase gradually when reducing the channel count from 32 to 8, indicating that the model retains sufficient representational capacity in this range. However, a sharp degradation in angular accuracy is observed when further reducing the number of channels to 4, suggesting a loss of directional selectivity and insufficient capacity to encode motion information.

These results highlight a clear trade-off between model compactness and accuracy, identifying 8 hidden channels as a lower bound for maintaining acceptable performance in dense event-based optical flow estimation.

### 5.2.2. Effect of Spatial Resolution Variation

The spatial resolution of input event frames is progressively reduced from 256×256 to 128×128, 64×64, and 32×32 pixels, by means of average pooling.

| Resolution | N. ops [M MAC] | Mem. [MB] | AEE ↓ [pixel] | AAE ↓ [rad - degree] |
|---|---|---|---|---|
| 256x256 | 7350 | 76.7 | 2.70 | 0.41 - 24 |
| 128x128 | 1837 | 19.3 | 2.44 | 0.52 - 30 |
| 64x64 | 459 | 5.0 | 2.68 | 0.86 - 49 |
| 32x32 | 114 | 1.4 | 2.79 | 1.13 - 65 |

Table 5.3.: Accuracy, Operations and Static Memory Allocation vs. Input resolution for the baseline model (32 channels), averaged over 4 MVSEC seq. and 2 temporal resolutions

To ensure metric coherence across resolutions, both input event frames and ground-truth optical flow are downsampled using average pooling prior to evaluation.

Varying the input spatial resolution reveals a markedly different behavior between modulus-based and angular error metrics. The Average Endpoint Error remains relatively stable across resolutions and even slightly improves when reducing the resolution from 256×256 to 128×128 pixels. This suggests that moderate spatial downsampling, when performed consistently on both input events and ground truth, preserves motion magnitude information while significantly reducing computational cost.

In contrast, the Average Angular Error increases monotonically as the resolution decreases. This behavior reflects the higher sensitivity of angular metrics to spatial quantization and reduced event density, which lead to noisier direction estimates at lower resolutions. The divergence between AEE and AAE highlights the importance of using complementary metrics when evaluating event-based optical flow models, particularly in deployment-oriented scenarios involving aggressive resolution scaling.

From a hardware perspective, these results indicate that substantial reductions in memory footprint and computational complexity can be achieved through resolution scaling with only a limited impact on modulus accuracy, making it an effective strategy for ultra-low-power deployment.

### 5.2.3. Role of Recurrence and Network Depth

To understand the contribution of temporal integration, models with and without recurrent convolutional layers are compared.

| Recurrence | Mem. [MB] | AEE ↓ [pixel] | AAE ↓ [rad - degree] |
|---|---|---|---|
| yes | 76.7 | 2.70 | 0.41 - 24 |
| no | 59.9 | 3.38 | 0.92 - 53 |

Table 5.4.: Accuracy and Static Memory Allocation wrt Current Recurrence presence for the baseline model (32 channels) @ 256x256 res., averaged over 4 MVSEC seq. and 2 temporal resolutions

Removing current recurrence leads to a noticeable reduction in static memory allocation; however, this comes at the cost of a substantial degradation in optical flow accuracy. In particular, the Average Angular Error more than doubles when recurrence is removed, indicating a severe loss of directional consistency.

These results confirm that temporal integration through recurrent LIF layers plays a critical role in event-based optical flow estimation, especially for accurately capturing motion direction. Given the limited memory savings relative to the significant accuracy loss, recurrence removal was not pursued further in this work.

| N. Layers | Mem. [MB] | AEE ↓ [pixel] | AAE ↓ [rad - degree] |
|---|---|---|---|
| 8 | 76.7 | 2.70 | 0.41 - 24 |
| 6 | 59.9 | 2.86 | 0.50 - 29 |

Table 5.5.: Accuracy and Static Memory Allocation wrt N. Convolutional + LIF layers for the baseline model (32 channels) @ 256x256 res., averaged over 4 MVSEC seq. and 2 temporal resolutions

Reducing the depth of the network by removing two non-recurrent convolutional LIF layers yields a comparable reduction in static memory allocation while introducing a significantly smaller degradation in accuracy. Both AEE and AAE increase moderately, suggesting that the remaining layers retain sufficient representational capacity for motion estimation.

Compared to recurrence removal, depth reduction provides a more favorable accuracy–memory trade-off and was therefore retained as a viable optimization strategy for deployment-oriented configurations.

Although both recurrence removal and depth reduction result in similar memory savings, their impact on accuracy differs substantially. While removing recurrence severely degrades directional accuracy, reducing the number of non-recurrent layers preserves most of the model's performance. This comparison highlights the importance of temporal modeling in event-based optical flow and motivates the architectural choices adopted in the remainder of this work.

## 5.3. Quantization Results

Quantization is a key enabler for deployment on ultra-low-power platforms. Both post-training quantization (PTQ) and quantization-aware training (QAT) are evaluated under different configurations.

### 5.3.1. Post-Training Quantization

PTQ is applied to convolutional layers and LIF neuron parameters, including membrane voltage states. Proper calibration of the voltage dynamic range proves critical to preserving accuracy. Fake quantization is introduced to evaluate the performance drop in the Pytorch[2] environment: the convolutional layers are quantized using Brevitas[3], while the LIF layers take advantage of SNN Torch state quantizers.

| Data type | N. ops [M MAC] | Mem. [MB] | AEE ↓ [pixel] | AAE ↓ [rad - degree] |
|---|---|---|---|---|
| fp32 | 7350 | 76.7 | 2.70 | 0.41 - 24 |
| int8 | 7388 | 19.9 | 2.75 | 0.41 - 23 |

Table 5.6.: Accuracy, Operations and Static Memory Allocation before and after PTQ, for the baseline model (32 channels) @ 256x256 res., averaged over 4 MVSEC seq. and 2 temporal resolutions

[2]https://pytorch.org
[3]https://github.com/Xilinx/brevitas

This table confirms that, with proper calibration of LIF membrane voltage states, Post-Training Quantization introduces only a negligible degradation in optical flow accuracy while reducing static memory allocation by approximately a factor of four, which is valid for most architecture variations. The slight increase in the number of operations is due to the introduction of quantization, dequantization, and requantization blocks in the quantized ONNX graph and does not affect the overall deployment feasibility. In some cases, quantization slightly improves performance, likely due to reduced saturation effects in LIF membrane voltages.

A critical aspect of Post-Training Quantization for spiking neural networks is the calibration of LIF neuron states. The default quantization behavior provided by the SNN Torch library applies a fixed and symmetric quantization range, typically limited to approximately $[-1, 1.5]$. While suitable for normalized activations, this range is insufficient for LIF membrane voltages, which can assume large negative values during inference.

Using the default calibration, negative voltage values are effectively clipped, severely limiting the neuron dynamics and resulting in substantial accuracy degradation. To address this issue, a per-layer statistical analysis of membrane voltage distributions is introduced. During calibration, voltage statistics are collected for each LIF layer and used to define a more appropriate quantization range.

Since the target quantization format is int8, an excessively wide range would reduce effective precision around the firing threshold. To balance dynamic range and precision, the lower bound of the voltage range is clipped at approximately $-150$. This choice preserves the majority of the voltage distribution while maintaining sufficient resolution near the threshold value.

To simplify the hardware implementation, only **uniform, per-tensor quantization** is employed. While more advanced schemes such as threshold-centered, non-uniform, or per-channel quantization would likely further improve accuracy, they were deliberately avoided due to their increased hardware complexity.

### 5.3.2. Quantization-Aware Training

QAT is explored to further reduce quantization-induced accuracy degradation. Several configurations are tested, including mixed-precision and fully quantized models.

Despite extensive tuning, QAT often suffers from training instability and vanishing gradients, particularly when quantizing recurrent LIF states. As a result, PTQ is favored as the most reliable and effective strategy within the scope of this work.

## 5.4. Hardware-Oriented Simulation Results

Quantized and floating-point models are simulated using Deeploy on PULP-based platforms to assess deployability.

All hardware-oriented simulations reported in this section are conducted targeting the Siracusa PULP platform. The simulated architecture features a hierarchical on-chip memory composed of 128 kB of L1 memory, 1.5 MB of L2 memory, and 32 MB of L3 memory, and operates at a clock frequency of 370 MHz. Unless otherwise stated, all memory allocation and performance results in this section refer to execution on this platform configuration.

These specifications define the memory and performance constraints under which the spiking neural network models are evaluated and directly influence tiling strategies, peak memory allocation, and achievable inference throughput.

### 5.4.1. Memory Allocation and Tiling

Static and dynamic memory allocations are measured during inference, accounting for neuron states, intermediate activations, and tiling buffers introduced by the execution backend.

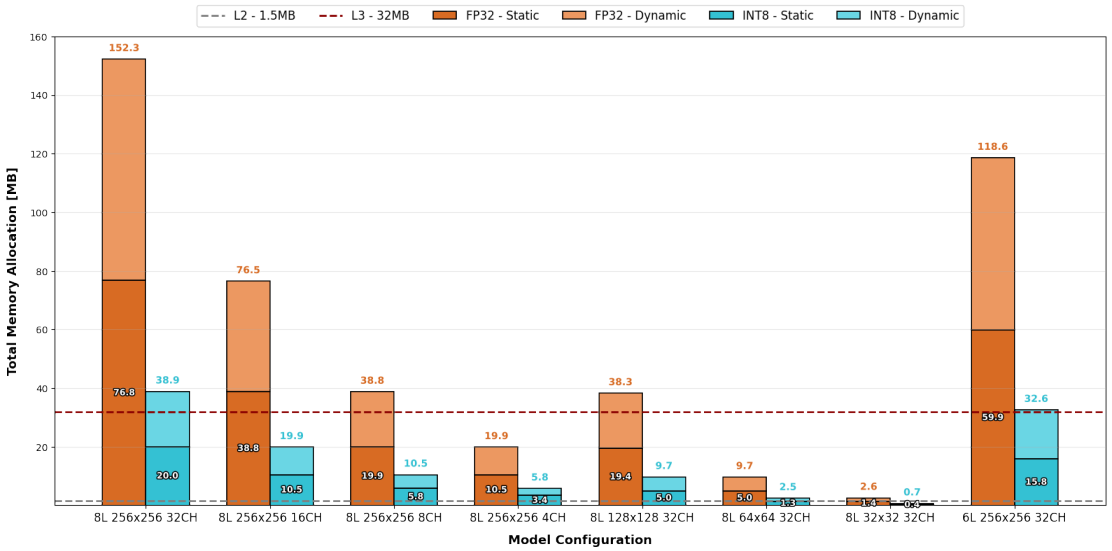


Figure 5.1.: Static and dynamic memory allocation for representative model configurations, comparing floating-point and int8 execution. Dashed lines indicate L2 (1.5 MB) and L3 (32 MB) memory limits of the target platform.

Figure 5.1 reports both static and dynamic memory allocation for a set of representative model configurations. Static memory accounts for model parameters and persistent LIF neuron states, which must be resident in memory throughout inference. Dynamic memory represents the aggregated allocation associated with intermediate activations, buffers, and tiling workspaces introduced by the execution backend.

While static and dynamic memory allocations are often of comparable magnitude, the reported dynamic value does not necessarily correspond to memory that must be allocated simultaneously. However, a non-negligible portion of the dynamic allocation must be available at runtime to enable correct execution. As a consequence, configurations whose static memory footprint fits within L3 may still be infeasible to execute entirely from L3 if the required dynamic working set exceeds the available capacity.

Post-Training Quantization remains a key enabler for deployability, substantially reducing static memory requirements and enabling execution of models that would otherwise exceed on-chip limits. In particular, several compact and quantized configurations fit entirely within L2 memory, eliminating the need for L3 accesses and representing the most favorable deployment scenario in terms of latency and energy efficiency.

Channel reduction, resolution scaling, and limited depth reduction further contribute to lowering memory requirements, while preserving recurrent connections consistently yields better accuracy for comparable memory footprints.

### 5.4.2. Computational Cost and Throughput

The number of operations, approximated as MACs, and the achievable inference throughput are evaluated for each model.

The computational cost of the explored models is approximated in terms of multiply–accumulate operations (MACs), while the achievable inference throughput is obtained from Deeploy simulations on the Siracusa platform. Throughput is reported as the maximum sustainable inference frequency, assuming a target clock frequency of 370 MHz.

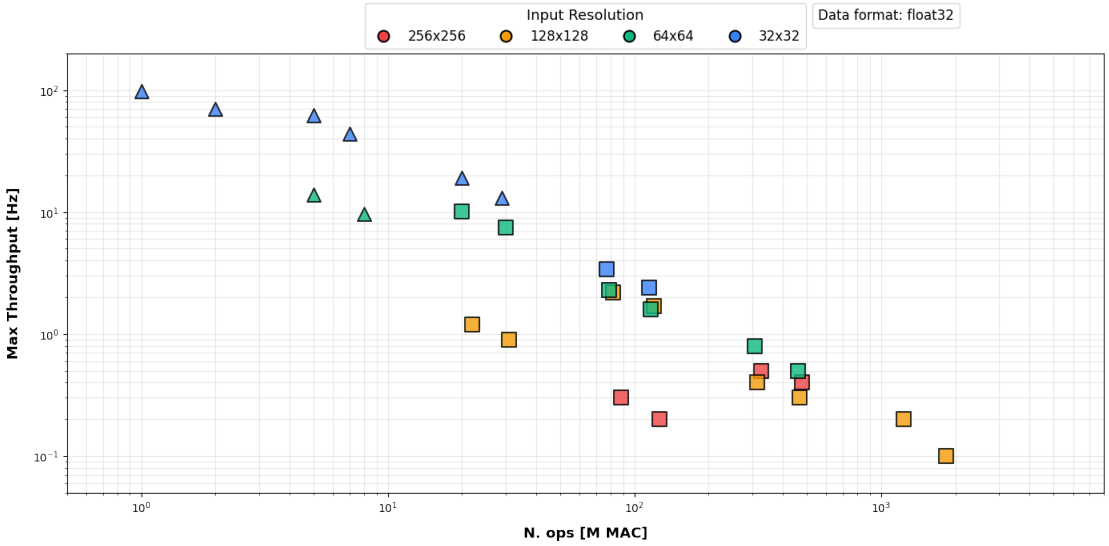


Figure 5.2.: Throughput wrt model complexity at fp32 precision: triangles indicate models executed from L2 memory, squares model executed from L3, both with tiling to L1

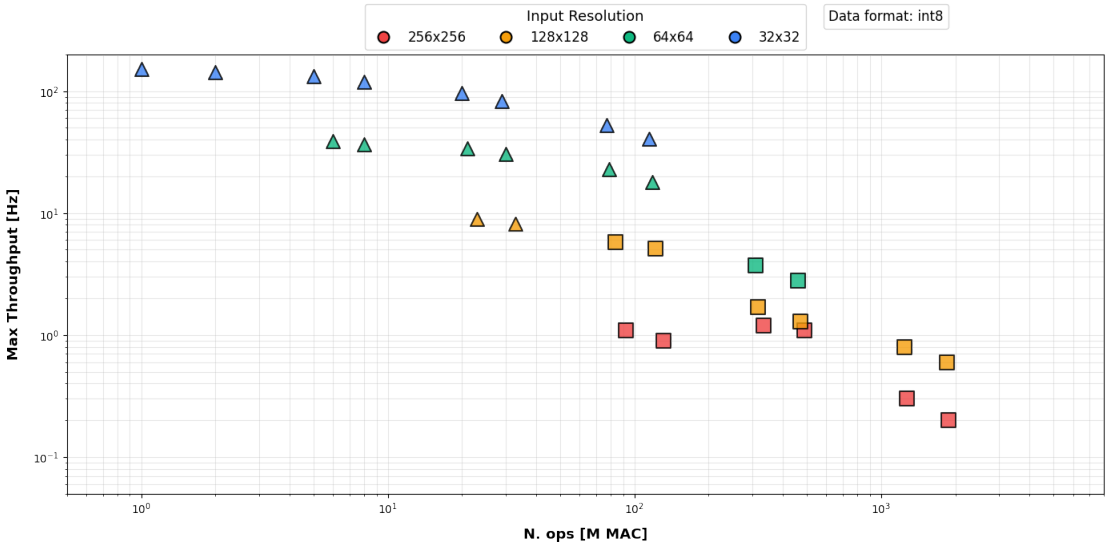


Figure 5.3.: Throughput wrt model complexity at int8 precision: triangles indicate models executed from L2 memory, squares model executed from L3, both with tiling to L1

Figures 5.2 and 5.3 summarize the throughput–complexity trade-off for all model configurations that could be successfully simulated, at floating-point (fp32) and integer (int8)

precision, respectively. Each point corresponds to a different combination of input resolution, number of channels, and network depth.

Triangles indicate models whose execution fits entirely within L2 memory, while squares represent models requiring L3 memory access, with tiling enabled between L3–L2 and L2–L1. Some model configurations are not shown in the plots, as they could not be simulated due to memory constraints—either because the static allocation exceeded available on-chip memory or because the dynamic memory requirements during execution could not be satisfied under the current Deeploy limitations.

Across all evaluated configurations, the achievable throughput spans more than three orders of magnitude. The smallest quantized models, operating at low spatial resolution and reduced channel count, reach peak inference rates of up to approximately 150 Hz, enabling real-time operation even under conservative execution assumptions. At the opposite end of the spectrum, large floating-point models operating at high resolution achieve inference rates as low as 0.1 Hz, which are impractical for real-time deployment.

A clear separation emerges between L2-resident and L3-resident models. Execution from L2 memory consistently yields significantly higher throughput, while models requiring L3 access suffer from substantial performance degradation. This effect is particularly pronounced for fp32 models, where memory bandwidth dominates execution time.

It is important to note that all throughput measurements represent a worst-case execution scenario, as they assume a single-buffer execution model. Due to current limitations in Deeploy, double buffering between memory levels is not enabled. In a real hardware implementation, overlapping data transfers with computation would substantially reduce the effective latency of L3-resident models, narrowing the performance gap between L2 and L3 execution.

Nevertheless, the presented results provide a conservative and hardware-aware estimate of achievable performance. They clearly demonstrate that model compactness, resolution reduction, and quantization are decisive factors in enabling real-time inference on ultra-low-power platforms. In particular, int8 quantization not only reduces memory usage but also expands the set of configurations that can operate at practical inference rates within on-chip memory constraints.

### 5.4.3. Simulation Constraints and Toolchain Limitations

While Deeploy enables detailed, memory-aware and cycle-accurate simulation of neural network execution on PULP platforms, some limitations of the current toolchain must be acknowledged.

First, **L3 tiling is currently affected by numerical inaccuracies**. Although memory allocation and latency estimates remain plausible, inference outputs obtained when enabling L3 tiling are not numerically reliable. As a consequence, results involving L3

execution should be interpreted primarily in terms of memory usage and performance trends rather than exact numerical accuracy.

Second, **double buffering is not yet supported** in the Deeploy execution backend. This limitation leads to pessimistic latency estimates when executing models from L3 memory, as data transfers and computation cannot be overlapped. In a real hardware implementation, double buffering would significantly reduce the performance gap between L2- and L3-based execution.

Despite these limitations, Deeploy remains a valuable tool for comparative analysis, as it accurately captures relative trends in memory footprint, computational cost, and throughput across different model configurations.

## 5.5. Summary of Results

The experimental evaluation highlights the following key findings:

- Spiking FireNet models provide a strong accuracy-efficiency trade-off for event-based optical flow estimation.
- Channel and resolution reduction are effective strategies for reducing memory usage with limited accuracy degradation.
- Recurrent LIF layers play a critical role in handling larger motion magnitudes.
- Post-training quantization enables substantial memory savings with minimal impact on accuracy.
- Hardware-oriented simulation confirms the feasibility of deploying optimized SNN models on PULP-based platforms.

These results collectively demonstrate that careful architectural design and quantization enable spiking neural networks to perform dense event-based optical flow estimation within the constraints of ultra-low-power embedded hardware.

Chapter 6

# Discussion and Conclusions

This chapter analyzes the results obtained in this thesis, discusses the main limitations of the proposed approach, compares the outcomes with existing state-of-the-art solutions, and summarizes the overall contributions. Finally, possible directions for future work are outlined.

## 6.1. Analysis of Results

The results presented in Chapter 5 demonstrate that spiking neural networks can effectively perform dense optical flow estimation from event-based data while meeting the stringent memory and computational constraints of ultra-low-power hardware platforms [4].

The baseline LIF FireNet architecture achieves competitive accuracy when evaluated on the MVSEC dataset, validating both the self-supervised training pipeline and the correctness of the spiking implementation. Performance degradation observed under larger temporal windows ($gtflow\_dt4$) is consistent with the increased motion magnitude and confirms the sensitivity of optical flow estimation to temporal integration [4, 11].

Architectural exploration reveals that channel reduction and spatial resolution scaling are the most effective levers for reducing memory footprint and computational cost. In contrast, aggressive layer pruning or removal of recurrent connections leads to disproportionate accuracy loss, highlighting the importance of temporal integration in event-based optical flow estimation.

Quantization results further strengthen these findings. Post-Training Quantization enables a significant reduction in memory usage with minimal impact on accuracy, provided that the dynamic range of LIF membrane voltages is correctly calibrated. The observed

cases where quantized models slightly outperform their floating-point counterparts suggest that quantization can act as an implicit regularizer, preventing excessive saturation of neuron states [14].

Hardware-oriented simulation using Deeploy confirms that optimized spiking models can be executed within the memory constraints of PULP-based platforms while achieving inference rates compatible with real-time operation (tens to hundreds of Hz, depending on configuration). These results collectively indicate that spiking neural networks represent a viable solution for low-power event-based optical flow estimation when designed with deployment constraints in mind.

Although all results are obtained through simulation, the target deployment platform for this work is the **Siracusa PULP chip**. This makes the reported results highly indicative of real deployment behavior.

## 6.2. Limitations

Despite the promising results, several limitations must be acknowledged.

First, all hardware evaluations are conducted through **cycle-accurate and memory-aware simulation**, rather than execution on physical silicon. While the employed simulation tools accurately model memory hierarchies, tiling strategies, and computational cost, real hardware measurements would be required to fully validate latency, energy consumption, and throughput.

Second, the training procedure does not exploit long-term temporal dependencies, as backpropagation through time is intentionally avoided to limit memory usage and training complexity [11]. This choice simplifies training but may restrict the model's ability to capture longer temporal correlations present in event streams.

Third, Quantization-Aware Training remains challenging for recurrent spiking networks [13]. Although Post-Training Quantization proved effective, QAT often suffered from instability, vanishing gradients, and saturation effects, particularly when quantizing neuron states. More robust training strategies are required to fully exploit the potential of quantization during training.

Finally, the evaluation is limited to a subset of available datasets and scenarios. While MVSEC provides reliable ground truth, additional datasets and real-world experiments would further strengthen the generality of the conclusions.

## 6.3. Comparison with State-of-the-Art

Compared to ANN-based approaches for event-based optical flow estimation, the proposed spiking models achieve competitive accuracy while offering substantial advantages in terms of memory efficiency and suitability for event-driven computation [4, 7]. Unlike encoder–decoder architectures with millions of parameters, the explored SNN models maintain a compact footprint that is compatible with embedded deployment.

Relative to hybrid ANN–SNN architectures, this work demonstrates that fully spiking models can be trained and deployed without relying on dense ANN components, preserving the benefits of sparse, asynchronous computation [9, 10]. While hybrid approaches may simplify training, they compromise the energy efficiency potential of neuromorphic systems.

In contrast to prior SNN-based optical flow works that focus primarily on algorithmic feasibility, this thesis places a strong emphasis on hardware-aware design, quantization, and deployability [12]. The integration of LIF neurons into ONNX and Deeploy, together with detailed memory and performance profiling on PULP platforms, distinguishes this work from existing studies that stop at software-level evaluation.

## 6.4. Conclusions

This thesis investigated the feasibility of spiking neural networks for ultra-low-power event-based optical flow estimation, with a focus on end-to-end deployment considerations. Starting from a state-of-the-art SNN architecture, the work systematically explored architectural trade-offs, evaluation metrics, quantization strategies, and hardware-oriented execution.

The results demonstrate that carefully designed spiking architectures, combined with appropriate quantization and memory-aware optimization, can achieve accurate optical flow estimation within the constraints of embedded neuromorphic hardware. The successful integration of LIF neuron layers into a software–hardware toolchain further highlights the practicality of deploying SNNs beyond purely theoretical studies.

Overall, this work contributes toward bridging the gap between spiking neural network research and real-world, low-power event-based vision systems, demonstrating that dense motion estimation with SNNs is not only algorithmically feasible, but also practically deployable.

## 6.5. Future Work

Several directions for future research naturally emerge from this work.

A first and essential step is the execution of the proposed models on real hardware, such as PULP-based chips or other neuromorphic platforms, to validate energy consumption and latency measurements experimentally [5]. GAP9 PULP chip, which is already largely compatible with the Deeploy toolchain, would be a perfect candidate for real execution as it would enable direct measurement of energy consumption and latency, completing the transition from simulation to real hardware.

From an algorithmic perspective, incorporating backpropagation through time or other temporal learning mechanisms could improve performance under large motion displacements, albeit at increased training complexity. Exploring more advanced neuron models or adaptive temporal dynamics may further enhance accuracy.

Another promising direction concerns the introduction of normalization mechanisms within spiking layers. Normalization could reduce the dynamic range of membrane voltages, particularly large negative values, simplifying quantization and potentially improving both numerical stability and accuracy. Exploring normalization strategies tailored to recurrent spiking dynamics represents an interesting avenue for future research.

On the optimization side, improving Quantization-Aware Training for recurrent spiking networks remains an open challenge. Developing training strategies that stabilize gradients and properly handle neuron state quantization would allow tighter integration between training and deployment.

Finally, extending the approach to additional event-based perception tasks, such as depth estimation or ego-motion estimation, would further demonstrate the generality and applicability of the proposed framework.

Appendix A

# Further Results

Here I will provide a complete set of plots, showing all 32 model variations explored, both at fp32 and int8 precision, comparing their complexity (here expressed as the number of operations) wrt their accuracy (AEE and AAE).

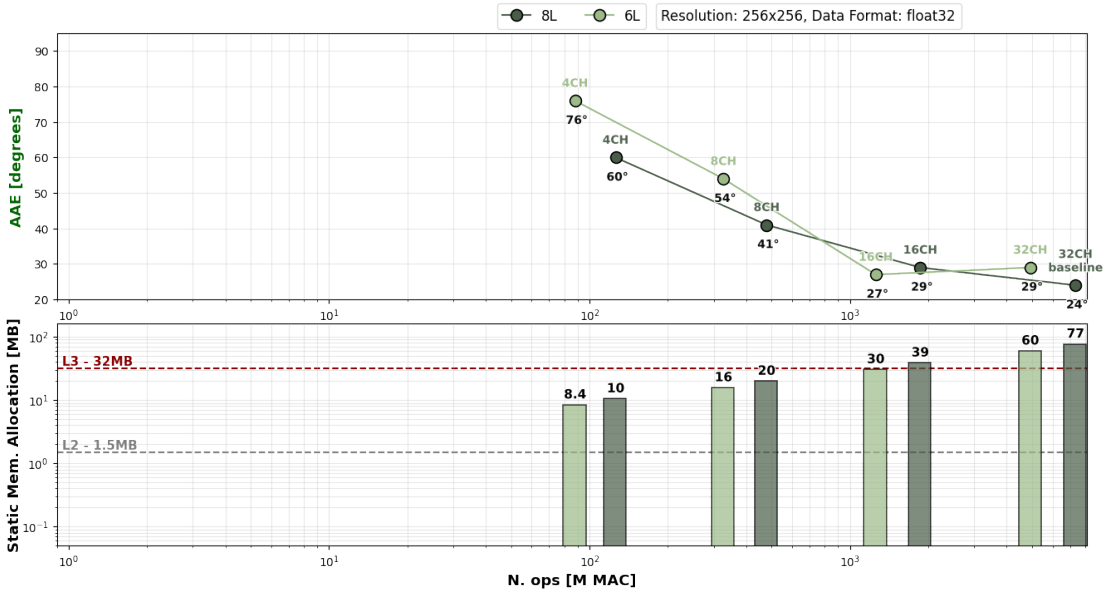


Figure A.1.: Average Angular Error (lower is better) and Static Memory Allocation wrt N. of operations [fp32, 256x256]

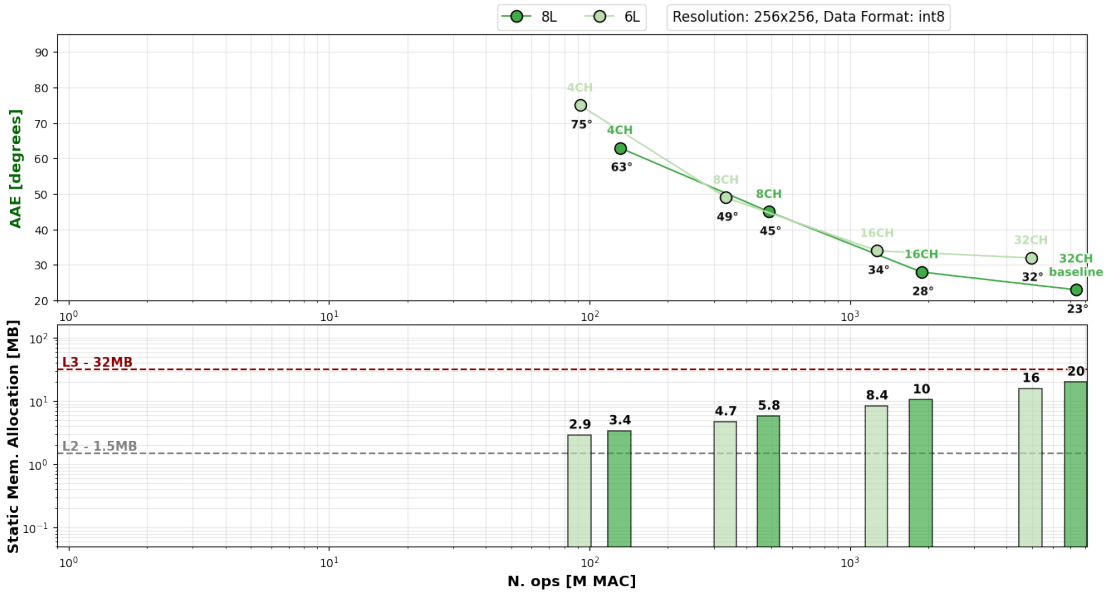


Figure A.2.: Average Angular Error (lower is better) and Static Memory Allocation wrt N. of operations [int8, 256x256]

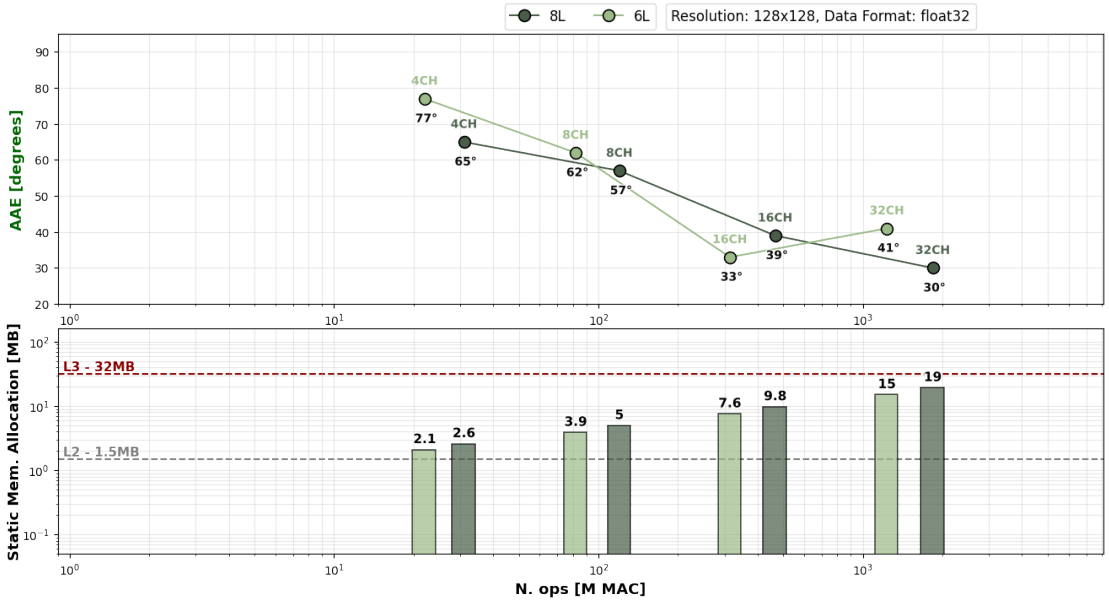


Figure A.3.: Average Angular Error (lower is better) and Static Memory Allocation wrt N. of operations [fp32, 128x128]

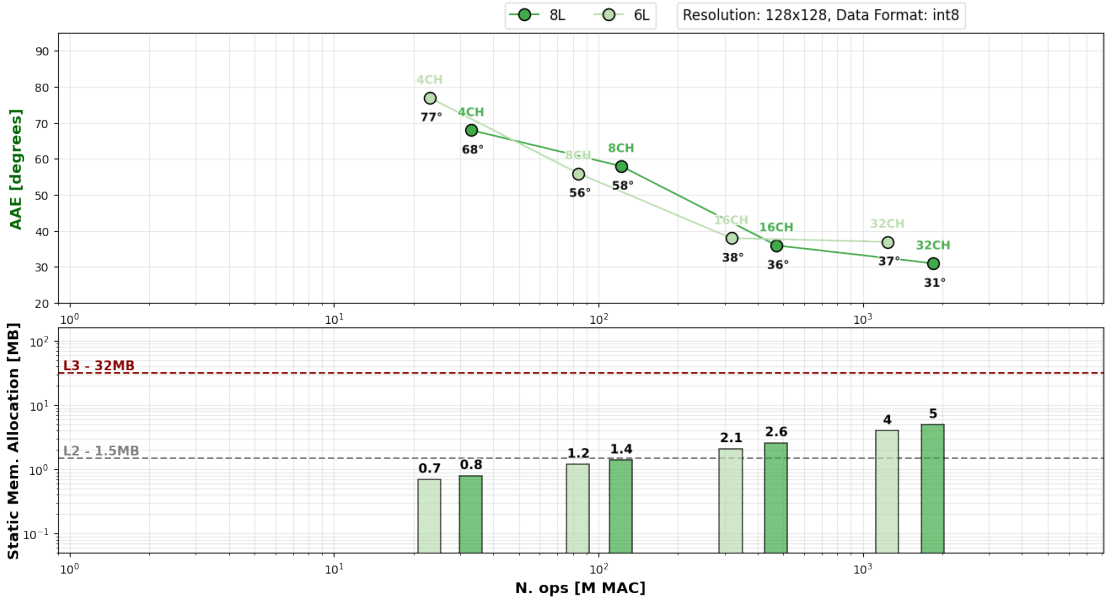


Figure A.4.: Average Angular Error (lower is better) and Static Memory Allocation wrt N. of operations [int8, 128x128]

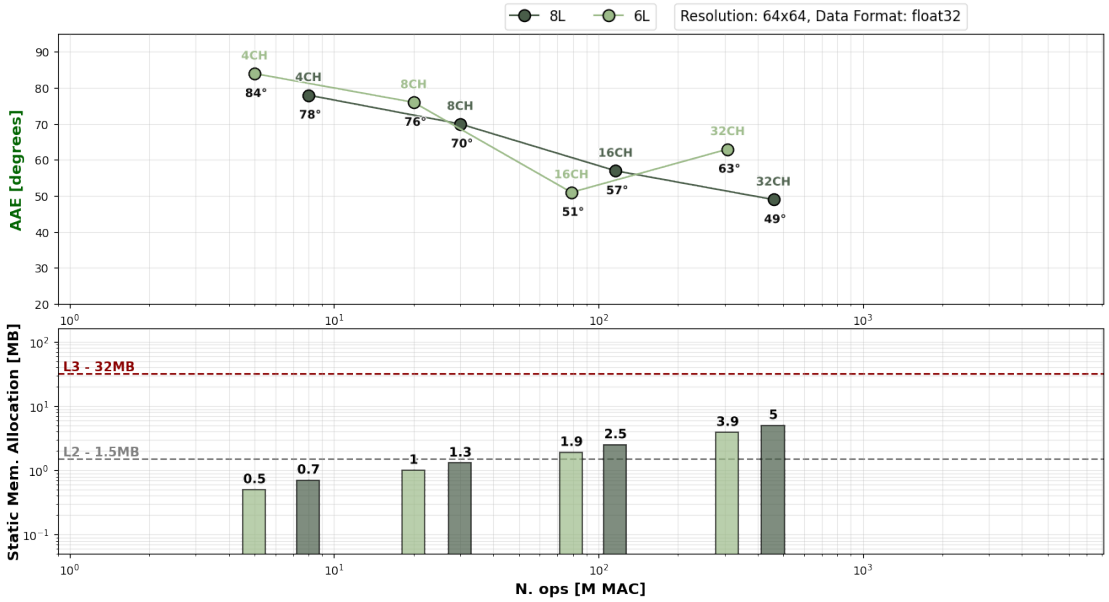


Figure A.5.: Average Angular Error (lower is better) and Static Memory Allocation wrt N. of operations [fp32, 64x64]

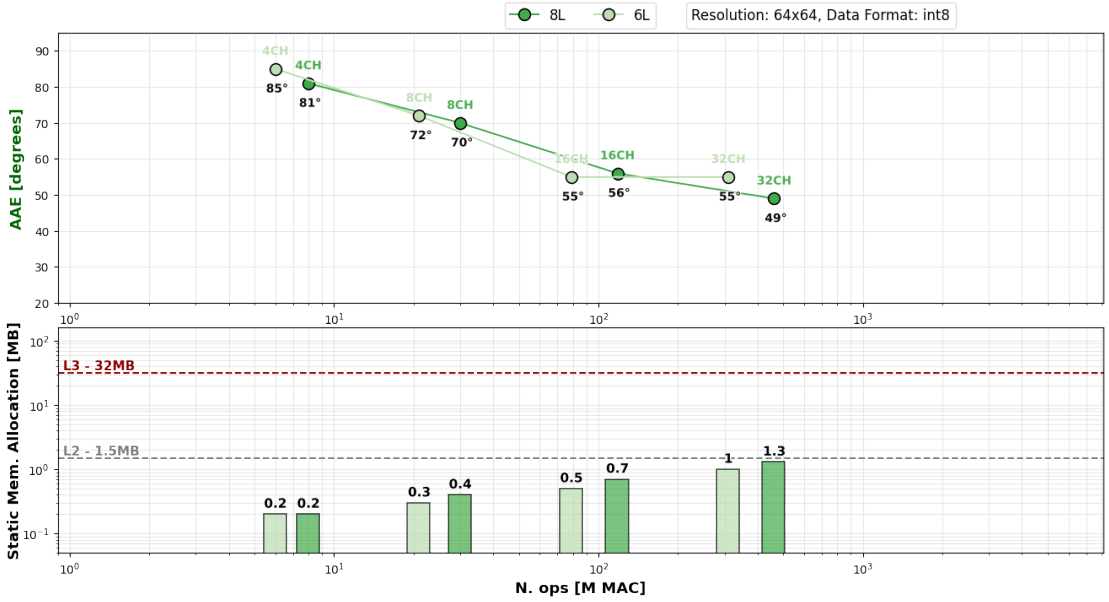


Figure A.6.: Average Angular Error (lower is better) and Static Memory Allocation wrt N. of operations [int8, 64x64]

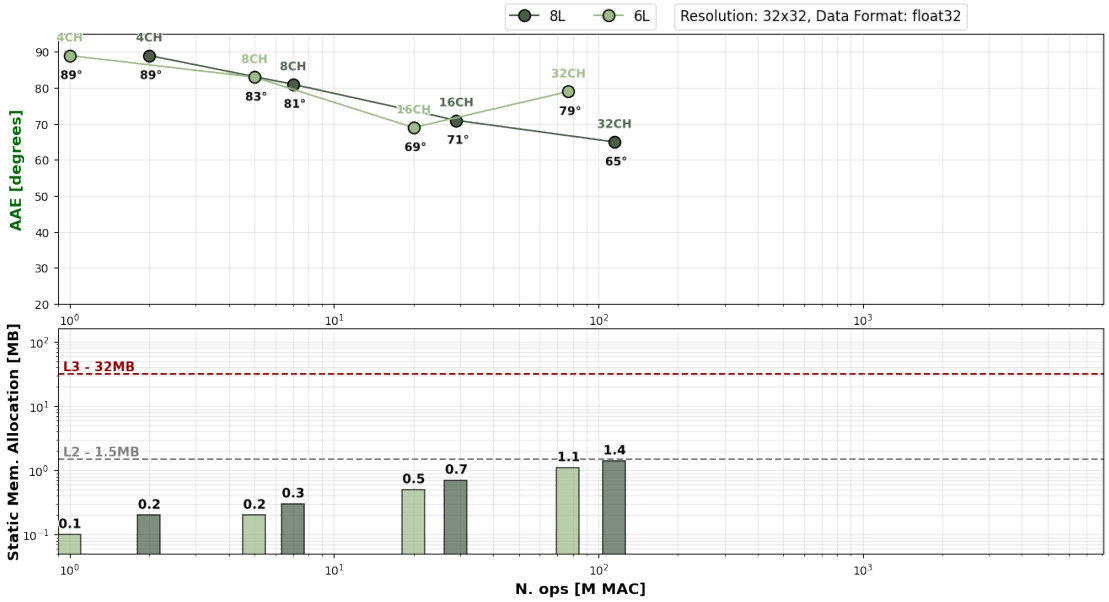


Figure A.7.: Average Angular Error (lower is better) and Static Memory Allocation wrt N. of operations [fp32, 32x32]

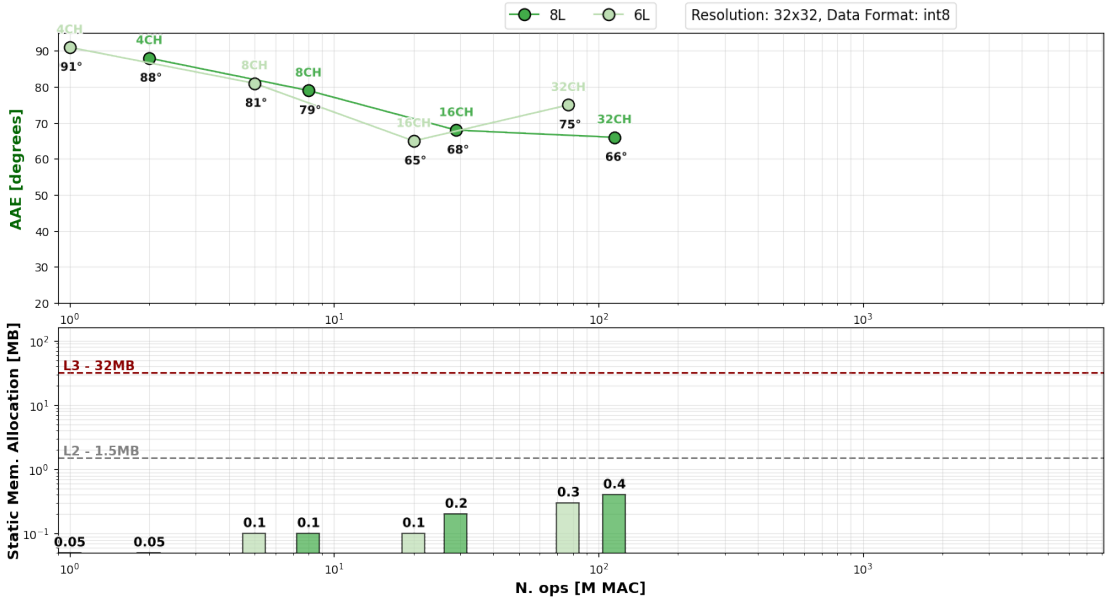


Figure A.8.: Average Angular Error (lower is better) and Static Memory Allocation wrt N. of operations [int8, 32x32]

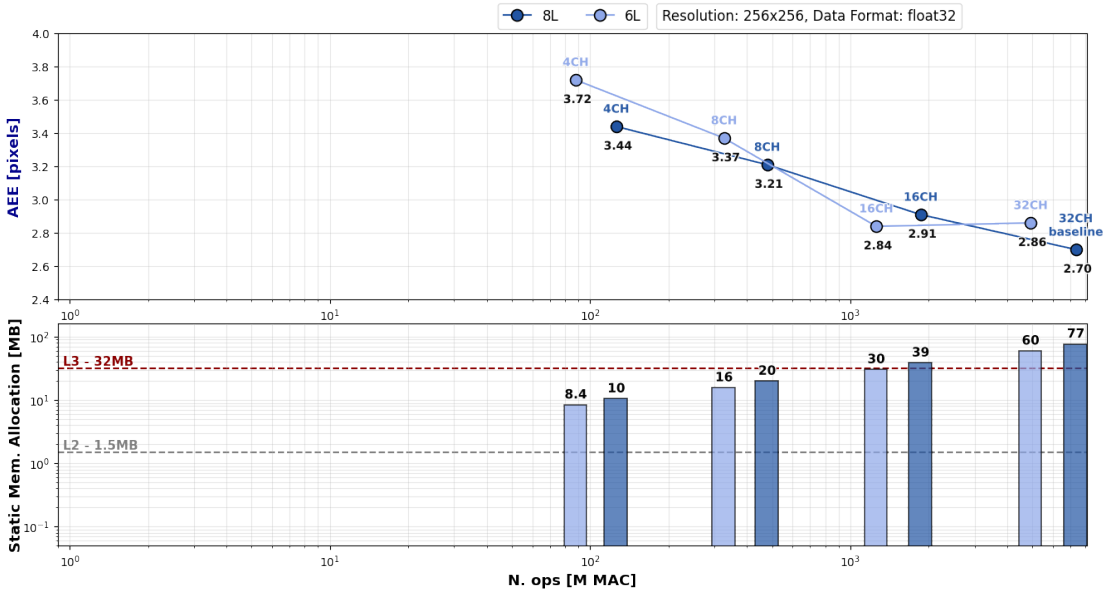


Figure A.9.: Average Endpoint Error (lower is better) and Static Memory Allocation wrt N. of operations [fp32, 256x256]

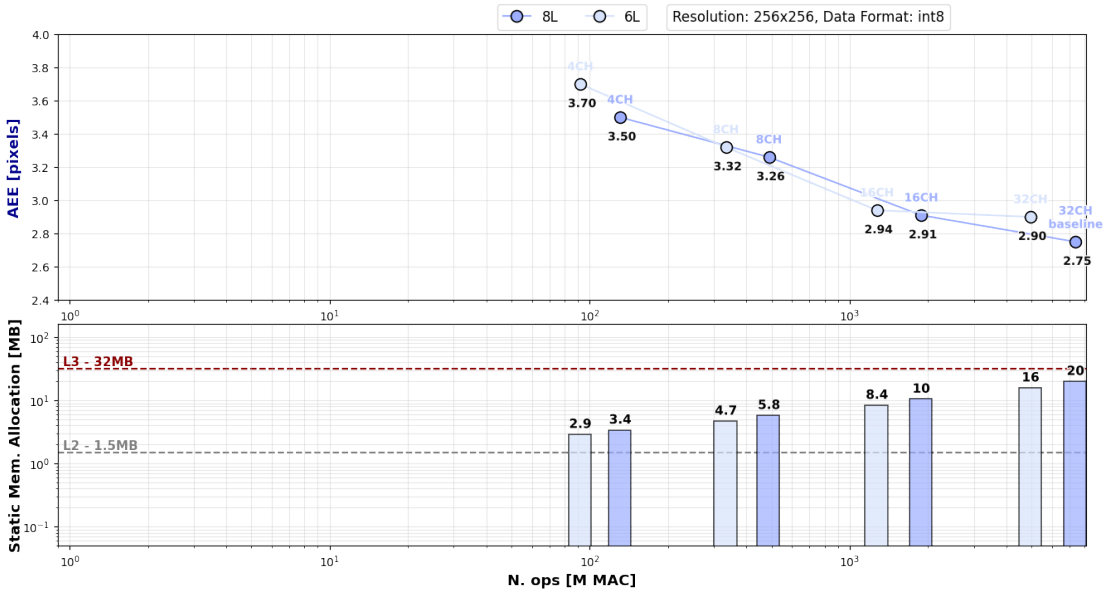


Figure A.10.: Average Endpoint Error (lower is better) and Static Memory Allocation wrt N. of operations [int8, 256x256]

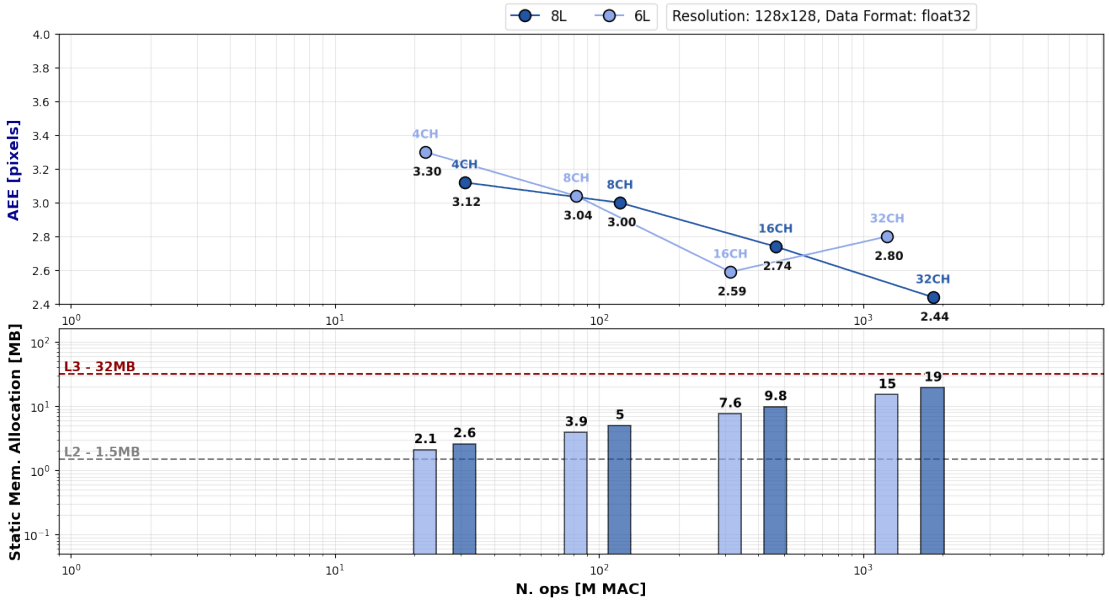


Figure A.11.: Average Endpoint Error (lower is better) and Static Memory Allocation wrt N. of operations [fp32, 128x128]

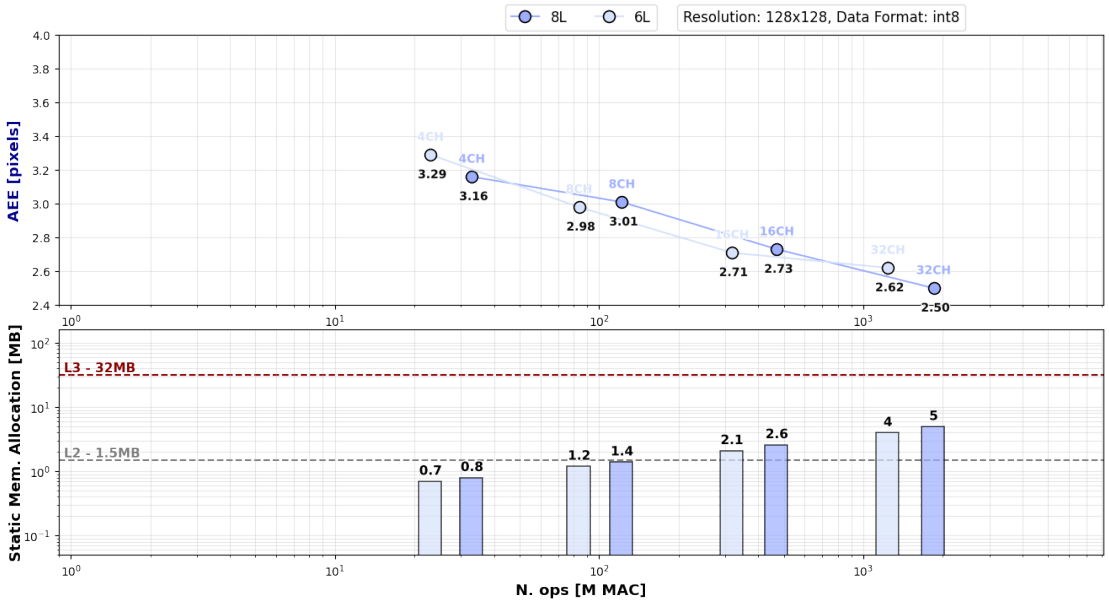


Figure A.12.: Average Endpoint Error (lower is better) and Static Memory Allocation wrt N. of operations [int8, 128x128]

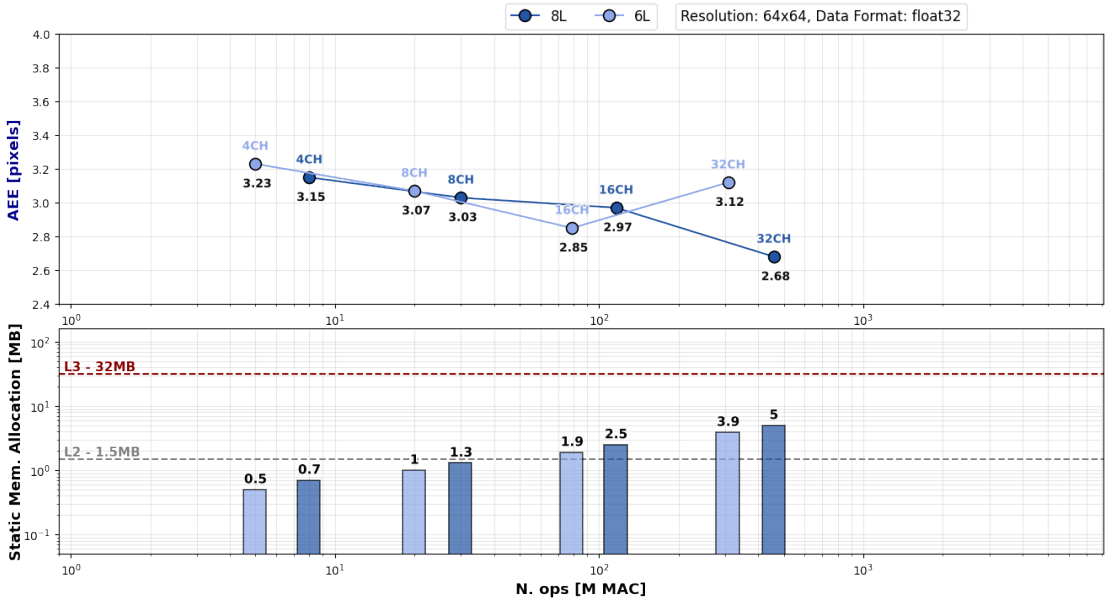


Figure A.13.: Average Endpoint Error (lower is better) and Static Memory Allocation wrt N. of operations [fp32, 64x64]

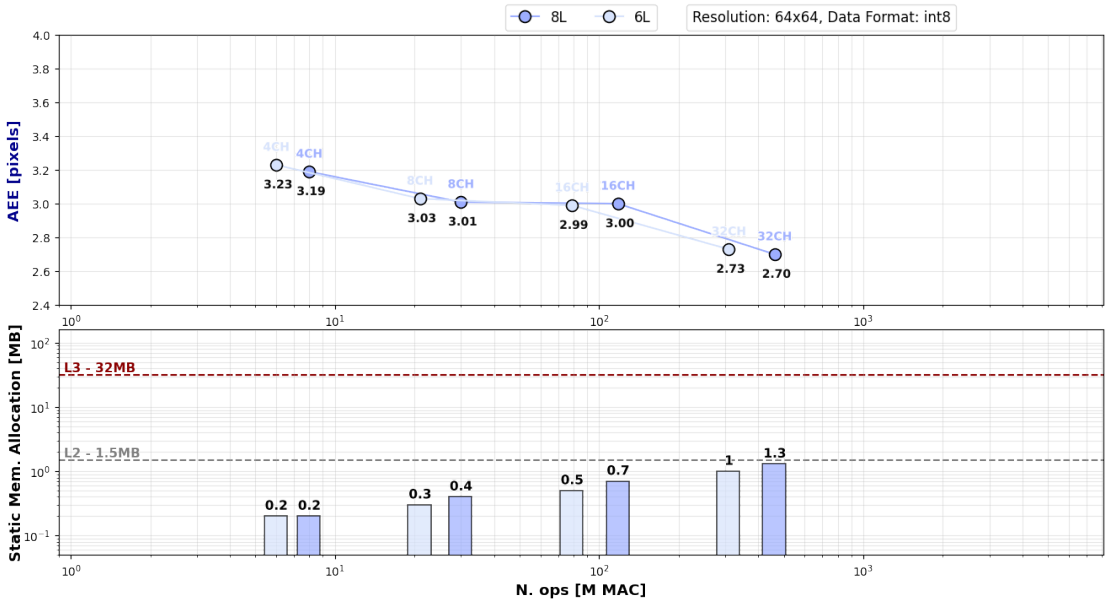


Figure A.14.: Average Endpoint Error (lower is better) and Static Memory Allocation wrt N. of operations [int8, 64x64]

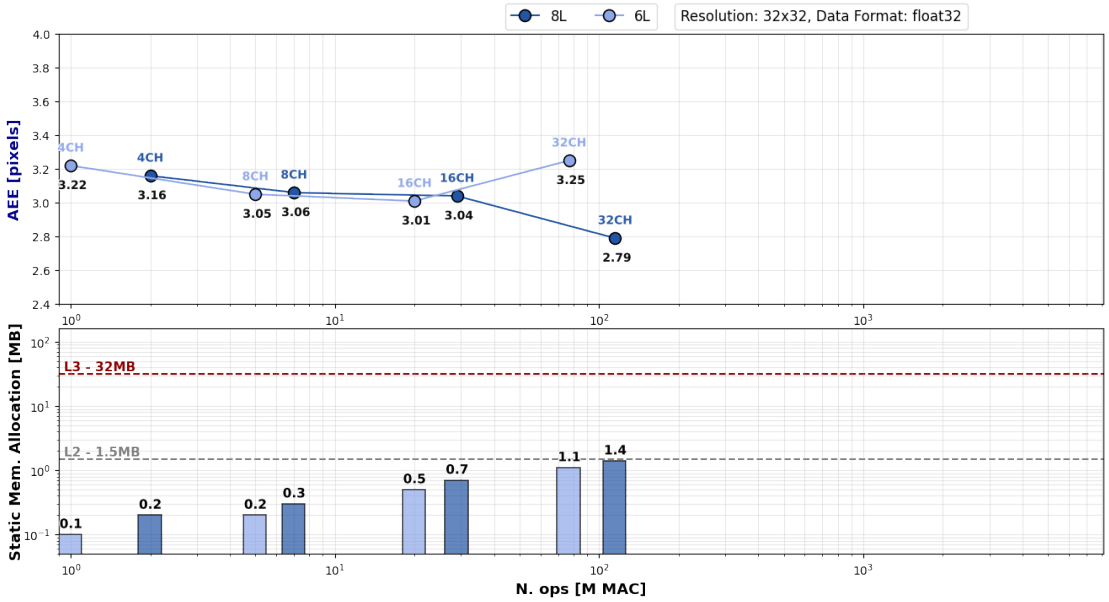


Figure A.15.: Average Endpoint Error (lower is better) and Static Memory Allocation wrt N. of operations [fp32, 32x32]

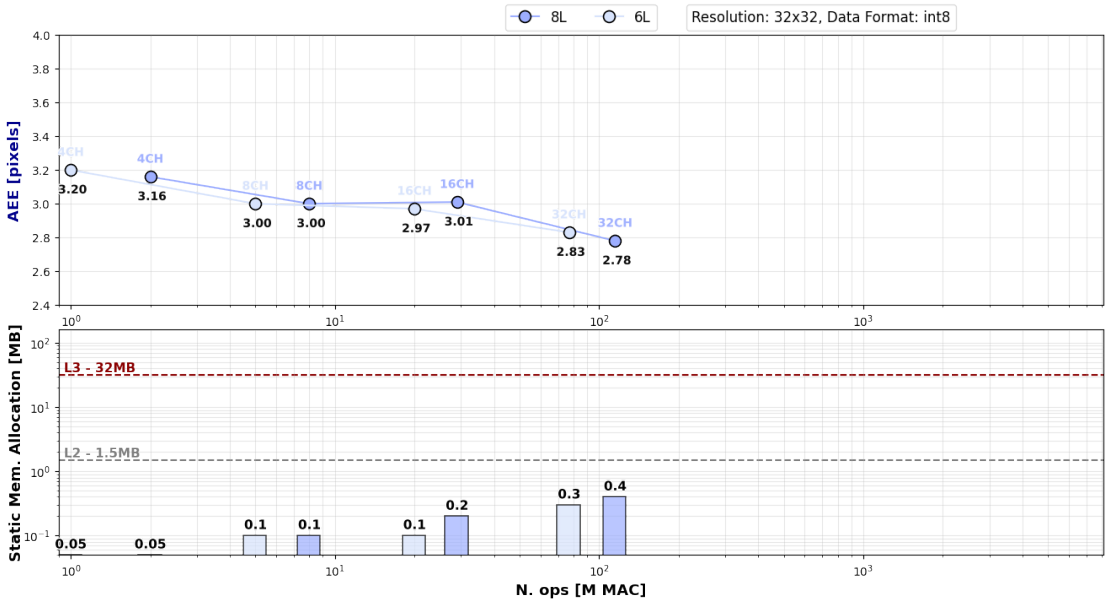


Figure A.16.: Average Endpoint Error (lower is better) and Static Memory Allocation wrt N. of operations [int8, 32x32]

# Bibliography


[1] G. Gallego, M. Gehrig, and D. Scaramuzza, "Focus is all you need: Loss functions for event-based vision," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun. 2019, p. 12272–12281. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2019.01256

[2] M. Scherer, L. Macan, V. Jung, P. Wiese, L. Bompani, A. Burrello, F. Conti, and L. Benini, "Deeploy: Enabling energy-efficient deployment of small language models on heterogeneous microcontrollers," 2024. [Online]. Available: https://arxiv.org/abs/2408.04413

[3] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, p. 154–180, Jan. 2022. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2020.3008413

[4] J. Hagenaars, F. Paredes-Vallés, and G. de Croon, "Self-supervised learning of event-based optical flow with spiking neural networks," 2021. [Online]. Available: https://arxiv.org/abs/2106.01862

[5] A. Shrestha, H. Fang, Z. Mei, D. P. Rider, Q. Wu, and Q. Qiu, "A survey on neuromorphic computing: Models and hardware," *IEEE Circuits and Systems Magazine*, vol. 22, no. 2, pp. 6–35, 2022.

[6] J. Cuadrado, U. Rançon, B. R. Cottereau, F. Barranco, and T. Masquelier, "Optical flow estimation from event-based cameras and spiking neural networks," *Frontiers in Neuroscience*, vol. 17, May 2023. [Online]. Available: http://dx.doi.org/10.3389/fnins.2023.1160034

[7] Y. Tian and J. Andrade-Cetto, "Sdformerflow: Spatiotemporal swin spikeformer for event-based optical flow estimation," 2024. [Online]. Available: https://arxiv.org/abs/2409.04082

[8] W. Ponghiran, C. M. Liyanagedera, and K. Roy, "Event-based temporally dense optical flow estimation with sequential learning," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 9793–9802.

[9] C. Lee, A. K. Kosta, and K. Roy, "Fusion-flownet: Energy-efficient optical flow estimation using sensor fusion and deep fused spiking-analog network architectures," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 6504–6510.

[10] S. Negi, D. Sharma, A. K. Kosta, and K. Roy, "Best of both worlds: Hybrid snn-ann architecture for event-based optical flow estimation," 2024. [Online]. Available: https://arxiv.org/abs/2306.02960

[11] C. Lee, A. K. Kosta, A. Z. Zhu, K. Chaney, K. Daniilidis, and K. Roy, "Spike-flownet: Event-based optical flow estimation with energy-efficient hybrid neural networks," 2020. [Online]. Available: https://arxiv.org/abs/2003.06696

[12] A. K. Kosta, A. Joshi, A. Roy, R. K. Manna, M. Nagaraj, and K. Roy, "Toffe – temporally-binned object flow from events for high-speed and energy-efficient object detection and tracking," 2025. [Online]. Available: https://arxiv.org/abs/2501.12482

[13] S. Venkatesh, R. Marinescu, and J. K. Eshraghian, "Squat: Stateful quantization-aware training in recurrent spiking neural networks," 2024. [Online]. Available: https://arxiv.org/abs/2404.19668

[14] W. Wei, M. Zhang, Z. Zhou, A. Belatreche, Y. Shan, Y. Liang, H. Cao, J. Zhang, and Y. Yang, "Qp-snn: Quantized and pruned spiking neural networks," 2025. [Online]. Available: https://arxiv.org/abs/2502.05905

[15] F. Paredes-Vallés and G. C. H. E. de Croon, "Back to event basics: Self-supervised learning of image reconstruction for event cameras via photometric constancy," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 3445–3454.

[16] C. Scheerlinck, H. Rebecq, D. Gehrig, N. Barnes, R. E. Mahony, and D. Scaramuzza, "Fast image reconstruction with an event camera," in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020, pp. 156–163.

[17] T. Alhersh, S. Belhaouari, and H. Stuckenschmidt, "Metrics performance analysis of optical flow," in *VISAPP*, ser. VISIGRAPP 2020 - Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, G. Farinella, P. Radeva, and J. Braz, Eds. SciTePress, 2020, pp. 749–758, publisher Copyright: Copyright © 2020 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved.; 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2020 ; Conference date: 27-02-2020 Through 29-02-2020.

[18] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, "Are we ready for autonomous drone racing? the uzh-fpv drone racing dataset," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6713–6719.

[19] A. Z. Zhu, D. Thakur, T. Ozaslan, B. Pfrommer, V. Kumar, and K. Daniilidis, "The multivehicle stereo event camera dataset: An event camera dataset for 3d perception," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, p. 2032–2039, Jul. 2018. [Online]. Available: http://dx.doi.org/10.1109/LRA.2018.2800793