POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Creation of a neuromorphic dataset for low-power and privacy-aware gesture recognition

Supervisors

Prof. Stefano DI CARLO

Prof. Alessandro SAVINO

Prof. Alessio CARPEGNA

Prof. Gianluca AMPRIMO

Candidate

Fabio QUAZZOLO

December 2024

Summary

In the context of telemedicine and telerehabilitation, activity recognition plays a crucial role, especially in camera-based systems.
For example, in Parkinson's disease, many camera-based systems for automatic hand motor impairment assessment and rehabilitation, require an initial step of gesture recognition. In addition, the privacy of these systems is crucial and traditional RGB videos can be associated with significant privacy issues, especially during data transmissions.
A new paradigm of cameras, such as event cameras, are promising solutions for anonymous gesture recognition. Event cameras are neuromorphic devices inspired by the human retina, acquiring only the information deemed useful, with a high temporal resolution (in the order of microseconds), low consumption and a high dynamic range. Combined with neuromorphic architectures such as spiking neural networks, these systems may provide solutions for gesture recognition at the edge, guaranteeing low consumption and compliance with privacy regulations.
This thesis aims at collecting a large-scale dataset for gesture recognition using an event camera (DVXplorer Lite), together with some traditional RGB-Depth devices (Azure Kinect) and a motion capture system (OptiTrack), for reference. This dataset can be exploited for training robust spiking neural networks for hand gesture recognition.
In further details, 25 subjects were recruited among the student population of Politecnico di Torino and more than 6 hours of hand gestures were recorded, with a 100 kHz sampling rate for events. Gestures were chosen from typical movements assessed in Parkinson's disease (e.g., finger tapping, hand opening-closing, pronation-supination), to provide a large benchmark of healthy subjects which may be exploited for further research in the field.
Finally, to provide an example of how this dataset may be exploited by future researchers, a first attempt at training a spiking neural network using the collected data was performed. Even if preliminary, results are encouraging and will provide indications for future iterations.

Acknowledgements

"L'ostacolo è la via"
E' il modo in cui descriverei il mio percorso al Politecnico.
Un cammino lento ed imperfetto ma necessario, fortemente voluto e soprattutto mio. Un maestro che mi ha insegnato molto. Ho imparato che attendere è diverso da aspettare, che il confronto è solo con se stessi e che prendersi il proprio tempo nelle cose non significa sprecarlo.
Lungo il cammino ho avuto il privilegio di conoscere persone come Stefano e Alessandro, docenti appassionati, figure ispiratrici ma soprattutto umane, capaci di creare un gruppo affiatato di talentuosi ragazzi come il Lab 6. Ricorderò con affetto, Gianluca e Alessio, brillanti ingegneri sempre disponibili a dare consigli e considerare le cose da un'altra prospettiva.
Riguardando indietro non posso non essere grato per tutte le persone che ne hanno sempre fatto parte. Ringrazio i Drocchetti, biondi e bruni, per avermi accolto in famiglia e mostrato l'importanza della disciplina e dell'avere fede. Ringrazio Gabriella per avermi accompagnato qui per la prima volta ed essere sempre una presenza costante sia nei momenti di gioia che in quelli difficili. Il maestro Galattico per avermi dato modo di sfogare le ansie e le paure, insegnandomi a giocare a tennis. Paoletta, Luisotta e le bimbe per i pranzetti dopo scuola, le cenette scintillanti e le belle vacanze vissute insieme. Gli amici di una vita Jack, Andreino, Matteino, Bobo, Dav, Sobre e Pitone siete stati una costante. Un grazie a Corins che mi ha trasmesso l'importanza e la bellezza di credere nei propri sogni. Lo, Enri e Sorti, doni preziosi del liceo che mi hanno insegnato che valgo molto più di un voto. Il mio amico Martin, per la sua grande voglia di vivere e per il suo esemplare *amor fati*. Un grazie a Mattia per avermi introdotto al mondo dei microcontrollori ed a Piesse per aver reso le sessioni di studio un po' più dolci.
Al mio migliore amico, compagno di avventure, da sempre figura di riferimento ed esempio di umiltà e dedizione.
A mia sorella per avermi insegnato ad abbracciare la sconfitta, per il suo supporto ed amore incondizionato e per aver creduto nella mia visione.
A chi non vedo, ma percepisco sempre con me

Siete la mia forza.

Table of Contents

List of Tables VIII

List of Figures IX

Acronyms XII

1 Introduction 1
- 1.1 Digital Healthcare and Telehealth . . . 1
- 1.2 Gesture Recognition in Telerehabilitation . . . 4
- 1.3 Research Objectives . . . 7

2 Background 8
- 2.1 Neurodegenerative Disorders . . . 8
- 2.2 Assessment and Rehabilitation of Parkinson Disease . . . 10
  - 2.2.1 The MDS-UPDR Scale . . . 10
  - 2.2.2 The Rehabilitation . . . 11
- 2.3 Computer Vision . . . 13
  - 2.3.1 Machine Learning . . . 14
  - 2.3.2 Convolutional Neural Networks . . . 16
  - 2.3.3 Privacy concerns in camera-based system . . . 19

3 Neuromorphic Systems 20
- 3.1 Neuromorphic Engineering . . . 20
  - 3.1.1 Neuromorphic Sensing . . . 21
  - 3.1.2 Neuromorphic Software . . . 22
  - 3.1.3 Neuromorphic Hardware . . . 23
- 3.2 Event Cameras . . . 25
- 3.3 Spiking Neural Network . . . 27
  - 3.3.1 LIF model . . . 27
  - 3.3.2 Spikes Encoding . . . 29
  - 3.3.3 SNN's learning approches . . . 31

3.4 Combining Event Camera and SNN . . . 33
3.5 Existing Neuromorphic Datasets . . . 33
3.5.1 N-MNIST . . . 34
3.5.2 DVS Gesture . . . 34

4 Dataset Creation 35
4.1 Multi-camera Setup . . . 35
4.1.1 DVXplorer Lite . . . 38
4.1.2 Kinect Azure . . . 38
4.1.3 OptiTrack . . . 39
4.1.4 Camera Calibration . . . 40
4.1.5 Camera Synchronization . . . 40
4.2 Experiment Protocol . . . 43
4.2.1 Tasks . . . 44
4.3 Data Collection Process . . . 45
4.4 Dataset Organization . . . 46

5 Training SNN for Gesture Recognition 48
5.1 Data Preprocessing . . . 48
5.1.1 From aedat to numpy . . . 49
5.1.2 Recordings Split . . . 49
5.1.3 Dataset Split . . . 50
5.1.4 Tonic Transformations . . . 53
5.1.5 Applying Transformation and Caching . . . 53
5.2 Model Architecture . . . 55
5.2.1 Original Net . . . 55
5.2.2 Simplified Net . . . 56
5.2.3 Dynamic Net . . . 57
5.3 Training . . . 58
5.4 Validation of the Model . . . 59
5.4.1 Cross Validation . . . 59
5.5 Hyperparameters Optimization . . . 60

6 Experimental Results 64
6.1 Original Net Performance . . . 64
6.2 Simplified Net Performance . . . 65
6.3 Dynamic Net Performance . . . 67

7 Discussion 71
7.1 Limitations of the Study . . . 71
7.2 Privacy and Energy Efficiency . . . 72
7.3 Future Works . . . 72

7.4 Conclusion 73

Bibliography 74

List of Tables

1.1 Benefits of Telehealth 2
1.2 Telehealth and Telemedicine Challenges 3

2.1 Machine Learning Models and Applications. 16

3.1 Frame based vs Event Camera 27

4.1 Task combinations with different speeds (S, N, F), repeated twice for each combination (22 tasks per hand). 46

5.1 Distribution of subjects between Training Set and Test Set. 52
5.2 Original Net Structure. 56
5.3 Simplified Net Structure. 57
5.4 Dynamic Net Structure. 58

List of Figures

2.1 Convolutional Operation 17
2.2 Hierarchical recognition: classifying a bicycle 18
2.3 CNN basic structure 19

3.1 Frame vs Event based Cameras 26
3.2 Biological membrane(a). RC membrane model(b)[7] 28
3.3 Behavior of a LIF neuron [7] 29

4.1 Multicamera setup: DVXplorer Lite in the green circle, Azure Kinect in the blue ones. Esync in yellow for camera synchronization 36
4.2 Lab setup: subject position in orange, optritrack cameras in pink and operator positions indicated by blue arrows A and B. 37
4.3 GIF Checkerboard, inverting colours. 41
4.4 DVXplorer sync connectors 42
4.5 Different GNDs were welded to have the same reference 42
4.6 Marker Configuration: A for Finger Tapping, B for Finger to Nose, C for Open/Close, Tremor and Pronation/Supination. 44
4.7 Sequence of gestures represented by 4 frames each. All sequences are accumulated at 30 fps. 47

5.1 Lab luminosity and hand size per subject 51
5.2 Hand Sizes 51
5.3 Dispersion Graph with subject markers 52
5.4 Transformations sequence: original data, denoised, cropped, and downsampled (1 frame illustrated). 54
5.5 Hyperparameter optimization pipeline 63

6.1 Original Net Performance 65
6.2 Simplified Net Performance 66
6.3 500 Neurons Net 67
6.4 Dynamic Net Performance 68
6.5 Validation loss trend over epochs 69

6.6 Unfrozen Net with early stopping on underfitting runs 70
6.7 Promising run 70

Acronyms

AI
Artificial Intelligence

HCI
Human Computer Interaction

ML
Machine Learning

DNN
Deep Neural Network

ANN
Artificial Neural Network

PD
Parkinson's Disease

UPDRS
Unified Parkinson's Disease Rating Scale

VR
Virtual Reality

CNN
Convolutional Neural Network

SNN
Spiking Neural Network

LIF

Leaky Integrate and Fire Neuron

BPTT

Backpropagation Through Time

FT

Finger Tapping

NOSE

Finger To Nose

OC

Open Close Hand

TR

Tremor

PS

Pronation Supination

Chapter 1

Introduction

1.1 Digital Healthcare and Telehealth

The increase in chronic diseases, the aging of the population and, most notably, the COVID-19 pandemic, have highlighted the limits and problems of the traditional healthcare system, suggesting a new approach to medical practice based on the adoption of digital systems [1].
The rapid evolution of information and communication technologies has in fact led to the emergence of new models of digital healthcare that provide access to medical care remotely, breaking down geographical and temporal barriers. Among these there are "Telehealth" and "Telemedicine" but although they are often used interchangeably, they refer to slightly different fields [2].

Telemedicine can be defined as using telecommunications technologies to support the delivery of all kinds of medical, diagnostic and treatment-related services usually performed by doctors. For example, this includes conducting diagnostic tests, closely monitoring a patient's progress after treatment or therapy and facilitating access to specialists that are not located in the same place as the patient.
Telehealth is similar to Telemedicine but includes a wider variety of remote healthcare services beyond the doctor-patient relationship. It often involves services provided by nurses, pharmacists or social workers, for example, who help with patient health education and social support [2].
The introduction of these new models in the healthcare system has brought many benefits to both the patient and the service provider. For patients, especially those with reduced mobility, the elderly or living in remote locations, these technologies have made it possible to bring clinical services directly to their homes, thus avoiding costs in terms of time and travel and ensuring timely interventions.
In addition to reducing infrastructure and maintenance costs, this online approach

also allows healthcare workers to reduce exposure to pathogens, improving the safety of doctors and patients, especially during pandemics or lockdown periods [3]. Telemedicine platforms are also easily scalable, allowing to efficiently manage data from many patients, storing their medical history, including diagnoses, past medical prescriptions and progress over time, all accessible from a single application.

Patients as well as service providers are in favor of adopting and integrating

Benefits	**Description**
Increased Accessibility	Provides remote access to healthcare, especially beneficial for people in rural areas or with mobility impairments.
Cost Savings	Reduces the need for travel, saving time and reducing costs for patients.
Continuous Monitoring	Enables real-time monitoring of patients' health conditions via wearable devices or mobile apps.
Flexibility	Patients can schedule visits at their convenience with reduced wait times.
Improved Patient Engagement	Encourages patients to take an active role in their healthcare through access to online health data and remote consultations.

Table 1.1: Benefits of Telehealth

telemedicine into the healthcare system, for all the benefits mentioned above. However, its widespread adoption is still hampered by a number of barriers, including technology use among older adults, Internet bandwidth speeds in rural areas, and potential security breaches that could jeopardize patient privacy and confidentiality [2].

This last issue is particularly critical because it concerns strictly personal and sensitive patient's data that could be stolen and intercepted by malicious people who could then sell or alter them. It is therefore essential to build platforms that keep this data secure and at the same time develop encrypted communication protocols that guarantee the transmission of private and complete information.

This also involves ensuring the validity and accuracy of the information transferred, which may be compressed or reduced due to the limited bandwidth available. Incomplete information may lead healthcare professionals to make clinical decisions and treatment recommendations based on potentially inaccurate patient data that invalidate the effectiveness of the treatment itself.

Despite the still numerous challenges, Telehealth is expected to continue evolving as an integral part of routine care delivery.

Hybrid models combining in-person and virtual care could become more common,

allowing for more flexible and patient-centered approaches. Furthermore, advancements in artificial intelligence (AI), machine learning, computer vision and wearable technologies are expected to further enhance telemedicine services, particularly in the areas of diagnostics, telerehabilitation and personalized medicine.

Challenges	**Description**
Technological Barriers	Limited access to high-speed internet and lack of familiarity with digital devices can prevent the widespread use of telemedicine, especially in rural or low-income areas.
Data Privacy and Security Concerns	Ensuring the confidentiality and security of patient data is critical. Cybersecurity risks, including potential breaches of sensitive health information.
Lack of Standardization	The lack of unified telehealth platforms and interoperability issues between healthcare systems.
Regulatory Issues	Varying regulations policies across regions and healthcare systems can make it difficult for providers to offer telemedicine services consistently.
Patient Trust	Some patients, particularly older adults or those unfamiliar with technology, may be hesitant to trust virtual healthcare platforms or may struggle to use them effectively.

Table 1.2: Telehealth and Telemedicine Challenges

1.2 Gesture Recognition in Telerehabilitation

Gesture recognition is the process of identifying and interpreting human gestures, such as hand movements, through sensors or cameras.
This technology represents a fundamental piece in the new discipline of human-computer interaction (HCI), which deals with making human-machine communication as simple as possible but at the same time efficient and satisfying for the user. Movements are acquired by cameras or other sensors such as wearables and are processed by a computer that, based on the application context, will interpret and convert these into commands to interact with the software or control hand-freely the device.
The process usually requires the following steps:

1. *Data Acquisition*: images or videoframes, representing users's hand movements or body gestures, are acquired through cameras or wearble sensors.
2. *Pre-Processing*: the acquired data is pre-processed, to remove the noise or to improve the data quality.
3. *Feature Extraction*: relevant features are extracted from the pre-processed data.
4. *Gesture Recognition Algorithm*: ML algorithms are employed to analyze the extracted features and identify the specific gesture.
5. *Mapping to Commands*: recognized gestures are mapped to corresponding commands or actions.
6. *Feedback and Interaction*: once the gesture is recognized, the system will execute the corresponding command associated with the specific gesture.

Hand tracking and gesture recognition technology is revolutionizing entire industries from automotive to retail and customer service, from manufacturing to gaming and entertainment, improving the user experience in human-machine communication by facilitating contactless and intuitive interactions.

A sector that can greatly benefit from the application of this technology is modern healthcare. By taking advantage of the contactless nature of this technology, for example, surgeons can consult crucial medical records and diagnostic images without physically touching the interfaces with their hands, thus avoiding contamination during surgical operations.
In this context, gesture recognition and hand tracking can also be very valuable for remote monitoring and rehabilitation, especially in those cases where patients have particular motor deficits, allowing doctors to visit them and track improvements in

real time and without the need for physical presence.
A promising application could therefore be the assessment of hand impairment, which is an essential practice for monitoring neurodegenerative diseases, such as Parkinson's, over time [4].
This procedure is currently performed in presence and under the supervision of a physician. The patient is asked to perform certain tasks with both hands and the physician evaluates dexterity, range of motion, and any freezing to detect the severity and progression of the disease.
However, this in-person care, in addition to having shown all its vulnerabilities during the pandemic, may not be the best way to assess the patient's manual dexterity since it relies on short appointments, which are poorly assigned in outpatient clinics where patients behave differently than they would do at their home. In addition, neurologists assess disease progression using well-established rating scales and patient-reported questionnaires, which have interpretability problems and are subject to recall bias [5].

In this scenario, Telerehabilitation has emerged as a promising alternative.
Based on computer-vision solutions and ML algorithms, it has the potential to improve patient care and support physicians to manage PD more effectively.
By capturing and analyzing hand gestures through cameras, these systems are able to monitor the severity and progression of motor symptoms, such as tremors and rigidity, and evaluate the patient's response to therapy directly within their home environment and in an objective manner.
The data on the patient's hand gesture movements are indeed captured and then processed using ML algorithms that can classify different types of movements and assess their quality, possibly providing direct feedback to the patient.
This allows therapists to monitor patients' progress remotely and asynchronously and adapt rehabilitation plans accordingly. Likewise, this approach allows the patient to carry out their tasks comfortably from home, at any time of the day, without having to book a hospital visit and without long waiting times.
Besides, the patient is more involved in his rehabilitation path, having platforms available to consult his progress.

Telerehabilitation solutions, that use gesture recognition and machine learning algorithms, offer numerous benefits to Parkinson's patients and caregivers.
In addition to all the benefits of adopting telehealth solutions, it further provides an objective way to evaluate the patient's task performance, reducing the subjectivity that can arise from human assessments.

Despite the many benefits and great potential of this technology, there are still many challenges that limit its widespread adoption. Some of the most important

challenges to consider are:

- Data privacy and security.
- Power Consumption.

Indeed, the collection of remote sensitive health data raises concerns about patient privacy and personal information security. Many of the cameras usually used in image acquisition are common RGB sensors that capture detailed frames, including sensitive data regarding the patient's physical characteristics and surroundings (home, family, or even personal documents) which are not relevant for the analysis. Advanced encryption and secure data management protocols are therefore essential to ensure secure data transmission and storage.
The second challenge regards the energy demands required by RGB cameras and the computational power demanded for running common machine learning algorithm such as ANN which could become a critical issue if the adoption of these technologies becomes global.

In recent years, thanks to the enormous technological progress, new models of artificial neural networks and sensors have been developed that can process and acquire information by imitating the behavior of the human brain and eye, and perform these operations very efficiently and with minimal energy expenditure.
This new discipline is called Neuromorphic Engineering and promises great improvements in applications involving gesture recognition, object tracking and machine learning with a focus on privacy and low energy consumption.

1.3 Research Objectives

The advent of neuromorphic engineering has introduced new ways of acquiring and processing data with particular attention to the efficiency of these processes in terms of energy and computation.
In recent years, neuromorphic sensors have been developed for the first time, special cameras capable of acquiring events rather than classic frames.
Commonly called "event cameras", these sensors emulate the behavior of the human eye by recording only the information deemed useful with a high temporal resolution and a high dynamic range. These characteristics make them particularly suitable for applications related to gesture recognition, object tracking or artificial vision in the robotic field, as they are able to capture everything that "moves" in the order of microseconds and in very extreme lighting conditions.
Similarly, artificial neural networks evolved by drawing inspiration from the human brain and the way it processes information and learns.
Spiking Neural Network are based on a new model of neuron that, if and only if sufficiently stimulated, emits a spike, thus exploiting the advantages of the asynchronicity and sparsity of input information such as events.
The union of these technologies promises great improvements in terms of security, privacy of the acquired data and energy consumption in all those applications related to the tracking of hands or objects and their recognition.
Despite the growing interest in the event cameras field, there are still too few neuromorphic datasets available online with which to train SNN models, especially in the medical and rehabilitation domain.
The goal of this thesis is precisely the creation of a large-scale dataset of hand gestures typically used in the assessment of Parkinson's disease, using a particular neuromorphic sensor called DVXplorer-lite.
The dataset just created has been used to train and test a spiking neural network model to better understand the compatibility between the data produced by this type of sensor and the neuromophic model adopted.
The good results obtained want to be a starting point for future research studies and applications having neuromorphic systems at their center.

Chapter 2

Background

2.1 Neurodegenerative Disorders

In recent decades, thanks to scientific and technological discoveries in the medical and pharmaceutical fields, the average life expectancy has increased significantly, leading to a significant aging of the population.
These improvements have certainly brought many benefits since cures have been found for many diseases and technological machinery has let doctors to operate in a non-invasive, safe and very precise way on patients. The introduction of vaccines has also allowed to prevent many epidemics by blocking them at birth and containing their expansion.
All this has undoubtedly improved the general well-being of the population but at the same time the increase in life expectancy leads to the introduction of new diseases that are a natural consequence of aging.
Among the most common are Parkinson's and Alzheimer's, which currently affect millions of people around the world and whose incidence estimates are growing year after year.
Parkinson's disease (PD) is a progressive neurodegenerative disorder characterized by the gradual loss of dopaminergic neurons in the substantia nigra of the midbrain, which results in an unbalance between inhibitory and excitatory mechanisms. It is the second most common neurodegenerative disorder after Alzheimer's disease affecting 6 million people worldwide and it is predicted to be doubled by 2040 [6].
There is a lot of research going on to understand the causes of the disease, the possible correlations with a particular lifestyle or a possible genetic inheritance in order to develop an effective cure but, despite the many studies, we are still far from having a full understanding of it.
Current criteria used in the diagnosis of PD include the presence of bradykinesia (slowness of movements) as well as the presence of rest tremor in the hands, arms,

legs, jaw, and face but also difficulties with balance, speech, coordination and rigidity. These symptoms, which are considered "motor-symptoms" and are visible to the human eye, begin slowly and then gradually worsen over time.
On the other hand, even more important are those symptoms defined as "non-motor" and not easily visible, as they can occur many years before the disease is clearly diagnosed at a clinical level. Examples of these are depression, cognitive decline and sleep disorders.
Detecting these would be very crucial to diagnose PD at its inception, many years before the motor symptoms negatively affect the life of patients. Furthermore, it would be possible to start treatments that are able to limit or alter the normal course of the disease, although these "disease-modifying" treatments are not yet available.
Currently, there are no diagnostic tests available to detect Parkinson's disease before symptoms appear. Physicians assess disease progression in patients using a variety of clinical evaluations, often measured with the Unified Parkinson's Disease Rating Scale (UPDRS)[5].
The most commonly used treatment is *levodopa*, a drug that helps increase dopamine levels, temporarily alleviating symptoms. However, finding the right dosage of levodopa is complex, and as the disease advances, the drug's effectiveness diminishes (known as "wearing off").
At present, there is no cure to stop or reverse the progression of Parkinson's disease. Therefore, all available treatments aim to enhance patients' quality of life by managing symptoms.

2.2 Assessment and Rehabilitation of Parkinson Disease

Recent studies conducted by the Parkinson's Foundation Rehabilitation Medicine Task Force have shown that constant and continuous assessment and rehabilitation play a very important role in improving symptoms, functionality and quality of life and in reducing disability in patients with Parkinson's.
These practices, according to studies, are introduced into treatment only in the already advanced stages of the disease, when motor symptoms are already evident and limit the patient in daily activities.
According to the Parkinson's Foundation, the patient should instead undertake rehabilitation services in the early stages of the disease and constantly monitor its progress, modifying treatments based on its evolution.
The assessment therefore does not only serve to diagnose the disease, but plays a crucial role throughout the treatment and rehabilitation process.

2.2.1 The MDS-UPDR Scale

Assessment plays a key role in Parkinson's care, as periodic evaluations allow changes in the patient's condition to be monitored and the rehabilitation program to be adapted.
Assessment scales such as the MDS-Unified Parkinson's Disease Rating Scale (UPDRS), the Hoehn and Yahr Scale, or the Freezing of Gait Questionnaire (FOG-Q) are commonly used for this practice. The choice of scale depends on the objectives of the assessment, which can range from assessing motor symptoms to quality of life, non-motor symptoms, and functional risks, allowing doctors to create targeted treatment plans based on the situation.

The MDS-UPDRS is often preferred over other assessment scales because it provides a more complete and detailed picture of the condition of the patient with Parkinson's (PD). It is divided into four sections:

1. *Non-motor symptoms*: assesses cognitive, behavioral, and mood problems such as depression, anxiety, and other cognitive alterations.
2. *Activities of daily living*: to assess difficulty in performing daily activities, such as eating, dressing, and walking.
3. *Motor examination*: to measure motor symptoms such as tremor, muscle stiffness, bradykinesia (slowness of movement), and balance problems.
4. *Complications of therapy*: to evaluate side effects of medications such as dyskinesia and motor fluctuations.

The doctor guides the patient through all the phases of the experiment, assigning to each section of the questionnaire a score from 0 (no symptoms) to 4 (severe symptoms). All the scores of each section will be added together and will help to determine the overall level of disability of the patient.
This scale, as already mentioned, has been developed and validated through a rigorous process of international standardization, which involved experts and patients. The strengths, which make it the global standard used today, lie in its multidimensional nature as it provides an extremely detailed analysis of motor symptoms and not, unlike other scales that often focus on one dimension at a time.
In this thesis-project, some tasks, belonging to the analysis of the motor part of this scale, were taken into consideration as part of our experiment such as finger tapping, pronation and supination and tremor of the hand. These and other tasks will be introduced in more details in the following chapters.

2.2.2 The Rehabilitation

The rehabilitation process of patients with PD is complementary to the assessment and fundamental in the treatment and limitation of the disease, as it is able to improve the motor symptoms, mobility and balance of those who suffer from it.
The complexity of the disease requires a multidisciplinary approach that involves neurologists, physiotherapists, speech therapists, occupational therapists and psychologists, in order to address the many facets of this.
Among the most common practices we can mention:

- *Physical exercise*: muscle strengthening and aerobic exercises to increase strength, decrease stiffness and improve cardiovascular resistance. Stretching and balance and coordination exercises are also useful to improve balance and postural stability, reducing the risk of falls.

- *Occupational therapies*: all those activities that aim to promote health and well-being through occupation. The goal is to help patients improve the skills needed for common daily activities such as walking, writing, eating and more generally ensure safety and ease of movement in the home environment.

- *Speech Therapy Rehabilitation*: Difficulties in speaking and swallowing are common in patients with PD. Therefore, there are exercises to improve vocal tone, as well as exercises for safe swallowing to prevent the risk of choking.

In recent years, therapies based on virtual reality (VR) have been successfully tested through which patients are free to practice complex movements in complete safety and interact with visual and auditory stimuli in a protected environment.
These innovative therapies, better known as *"exergames"*, in addition to improving motor skills through specific exercises, exploit their "game" effect to increase the

patient's motivation and adherence to treatment through challenges, scores and goals to be achieved [6].
Many studies confirm that practicing sports or physical exercises in general help to increase brain plasticity by raising the levels of neurotrophic factors such as BDNF. To further enhance the benefits of physical activity, non-invasive therapies have been developed, based on transcranial direct current stimulation (tDCS) that use weak electrical currents to modulate brain activity.
The increasingly common use of wearable devices, such as motion sensors and biofeedback systems, allows for constant monitoring of the patient, providing objective data on progress and allowing for therapeutic program adaptation.
Furthermore, the large amount of data collected by these devices can be used to train complex algorithms and models to recognize possible patterns and help doctors predict the development of the disease.

2.3 Computer Vision

Computer vision is a branch of artificial intelligence that aims to enable computers to see, recognize, and understand the world around them, through images, videos, and other visual inputs.
To achieve this, computer vision uses algorithms and models based on machine learning that, when subjected to a large amount of data, are able to learn from it to identify relevant features and generalize to new data.
Among the most efficient and high-performance models are artificial neural networks, whose operation is directly inspired by the human brain. These networks are made up of several layers placed in sequence, each of which is composed of many neurons. These neurons can be modeled in different ways based on the desired behavior and the application that is to be implemented. Neurons are the basic units of an ANN, being able to exchange information between each other through interconnections called "synapses", to which are associated "weights" that indicate the importance of the connection.
In the following subchapters, some of these models will be explored in more detail and how they are able to understand and interpret an image will be explained.
These computer vision algorithms can analyze the image in various aspects, depending on the techniques used, the type of image and the task performed. Possible tasks include:

- *Image classification*: understanding which class the subject represented in the image belongs to, assigning it a label (example: it is a cat, it is a dog etc.).
- *Object detection*: using image classification to detect if one or more classes can be found in a given image (example: detecting the possible presence of damage on a surface, a machine or a product).
- *Image segmentation*: dividing the image into meaningful sections, called segments, to make image processing more efficient and faster.
- *Face Recognition*: recognition of people's faces for biometric security or automatic identification applications.
- *Action Recognition*: recognition of particular sequences of images that represent movements, signals, gestures. It is the basis of object tracking, that is, following an object, appropriately recognized, in space and time.

The ability to correctly perform these activities has opened the doors to multiple applications in as many sectors.
In the manufacturing and industrial sector, with the use of computer vision, it is possible to establish quality and recognize any defects in products. Similarly,

by analyzing images relating to the machinery itself and the production lines, it is possible to implement predictive maintenance mechanisms, avoiding machine downtime and delays in production. By monitoring the production environment, it is also possible to identify any risk situations that could compromise the safety of workers and prevent them from occurring.
In the automotive sector, more specifically in autonomous driving, artificial vision is essential for detecting obstacles, recognizing road signs and pedestrians. It allows the vehicle to accurately understand the surrounding environment and allows it to make safe decisions.
Finally, in medicine and telemedicine, it is useful in recognizing diseases such as tumors and other pathologies through the analysis of magnetic resonances and tomographies, allowing the doctor to operate with greater precision and adapt the therapy based on the patient's personal situation.
In addition to assessment and telerehabilitation applications, it is useful in recognizing gestures or other movements of interest to monitor the progress of the disease or simply to support the patient in his rehabilitation therapy by providing real-time feedback and helping him in exercises.
To achieve these goals, computer vision uses advanced techniques, often based on machine learning models that will be introduced in more detail in the following paragraphs.

2.3.1 Machine Learning

Machine Learning is a branch of artificial intelligence that aims to teach machines to learn something.
Unlike classic deterministic algorithms implemented by programmers by writing code, ML uses models and techniques that allow the algorithm to "learn" from the data itself provided as input.
There are various models capable of learning from data, each with its own characteristics and applications, summarized in Tab 2.1.
The model, appropriately chosen based on the desired task, is subjected to a large amount of data, such as images, videos, text or any other form of structured or unstructured information, this phase is called *Training*. During training, the model adapts its internal parameters in order to improve its predictions or decisions. This process is based on the optimization of a cost function that measures the discrepancy between the model's predictions and the actual input data, allowing the model to learn and improve based on its errors.
There are three main types of machine learning that differ in that they use different learning approaches:

- Supervised machine learning

- Unsupervised machine learning
- Reinforcement learning

Supervised machine learning uses rigorously labeled training datasets, which means that each input element to the model will be accompanied by its label representing the class to which the element belongs.
The model receives as input both the question (the data to be classified) and the correct answer to that question (the class to which it belongs). In this way, the model is constantly supervised during the training phase and, receiving the correct output, is able to modify its parameters to improve future predictions. This approach is particularly suitable for object classification problems.

In *unsupervised machine learning*, on the other hand, only the input (the data) is given to the model without providing it with any correct answers or labels. The model explores the data, analyzes it, and tries to group similar elements together. Based on these similarities in the data, the model then creates groups (called clusters) without knowing exactly what they represent. This type of approach is particularly useful when having large amounts of data that are difficult to analyze manually and not having labels for the data. Furthermore, the unsupervised nature is particularly advantageous for finding new patterns and correlations in the data, thus uncovering any hidden structures.

Reinforcement learning aims to make a model learn through trial and error mechanisms in a given environment, which involves rewards and penalties based on the resulting actions. Unlike the two approaches presented above, this type of learning is based on sequential decisions, where the action to be performed depends on the current state of the system and influences the future state. The agent behaves like a chess player during a game, each move and current situation of the environment influences future choices. This type of approach is particularly useful and promising in fields such as robotics where the environment is constantly evolving or in all those applications where it is important to optimize the long-term goal, such as finance and personalized marketing.

ML is the basis of many current applications, being able to exploit the large amount of data available to train models. Among these, it is particularly useful in Computer Vision because it allows computers to identify and interpret patterns in images and videos, tasks that would require complex programming, giving the model the ability to generalize to new inputs similar to those on which it was trained.
For this purpose, convolutional neural networks, a ML model, are very suitable due to their architecture proper to the recognition and processing of spatial patterns in

images. These will be briefly introduced in the following paragraph.

Model	Description	Application
DeepLearning Models	Models based on deep neural networks (ANN, CNN, RNN, SNN) capable of learning complex patterns.	Facial recognition, autonomous driving, medical diagnostics
Decision Trees and Random Forests	Sequential choice based models that divide data into segments; random forests combine multiple trees for greater accuracy.	Medical diagnosis, coded risk prediction, image classification
K-Nearest Neighbors (KNN)	Simple classification algorithm that assigns a label based on the most similar neighbors in the dataset.	Product recommendations, pattern recognition, image analysis
Support Vector Machines (SVM)	Algorithm that finds a hyperplane to separate categories of data, used for classification and regression problems.	Object recognition, text categorization, bioinformatics
K-Means Clustering	Unsupervised algorithm that groups data based on similarities, forming distinct clusters or groups.	Image segmentation, market analysis, anomaly detection

Table 2.1: Machine Learning Models and Applications.

2.3.2 Convolutional Neural Networks

Convolutional neural networks, abbreviated CNN, are a type of deep neural networks that are particularly capable of identifying patterns and recognizing images and objects.
Their deep nature is motivated by the presence of several hidden layers called "convolutional layers", responsible, as the name suggests, for convolution operations. The convolution operation involves the application of filters, called "kernels", to the pixel matrix representing the input image. More specifically, each filter is run over the input matrix, multiplying each value (of the filter) by the corresponding values in the portion of the image below and adding the result to obtain a single value for each position of the filter. This process, shown in Fig.2.1 generates a submatrix

called activation map (or feature map).
Each filter is responsible for detecting certain features, such as edges, and simple shapes for the first levels, up to increasingly complex and abstract features, such as entire objects, in the deeper layers.

This process of breaking down the problem into subproblems and recomposing

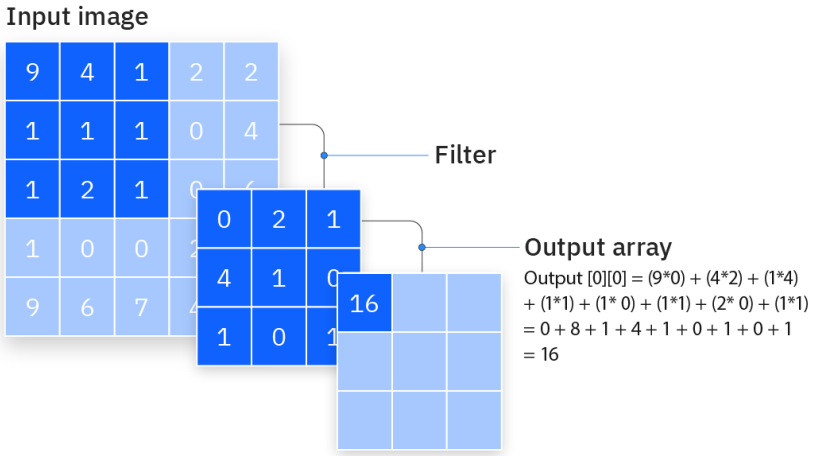


Figure 2.1: Convolutional Operation

the simple solutions to arrive at recognizing and classifying a given image, is metaphorically comparable to solving a puzzle. You start by identifying the tiles representing the edges of the various figures, you join these in order to give shape to the figures and gradually you proceed to join them to create the final image. A bottom-up approach as shown in Fig.2.2.

To reduce the dimensionality of the data produced, the so-called pooling layers are placed between the various convolutional layers, which have the objective of sub-sampling the feature maps by summarizing the information they represent.
The alternation between convolutional layers and pooling layers constitutes that part of the network dedicated to feature extraction.
Subsequently, the feature maps are flattened to form a single large one-dimensional vector, this phase is called "Flattening". At this point the vector is mapped into what is called the "Fully Connected Layer" in which all the neurons belonging to a layer are connected to all the other neurons of the previous and subsequent layers. This layer is typical of ANN networks and is responsible for the actual classification phase, at the end of which the network returns its prediction in the form of a percentage.
In order for the network to be able to learn complicated relationships and approximate complex functions, activation functions have been introduced, which are

Figure 2.2: Hierarchical recognition: classifying a bicycle

mathematical functions that introduce non-linearity in the model. Without them, each layer of the network would be just a linear combination of the inputs.
These functions are applied to the output of each neuron in all convolutional layers and in fully connected layers. There are many activation functions to choose from, which have specific characteristics that can influence the learning speed, the representation capacity of the network and the risk of some problems. Among the most used we must mention:

- *ReLU*: stands for "Rectified Linear Unit" and is the most used function in CNNs. It is defined as $f(x) = \max(0, x)$, so it only passes positive values and makes negative values zero.
- *Sigmoid and Tanh*: used in the simplest models, especially in the final layers.
- *Softmax*: typically used in the last layer of classification networks, it normalizes the output in a probability distribution for each class.

In general, the choice of the most suitable activation is guided by the complexity of the network and the task to be performed. Compared to Sigmoid and Softmax, which require exponential calculations, ReLU are very fast and easy to calculate and for this reason more commonly used.
In Fig.2.3 a summary diagram of the structure of a typical CNN.

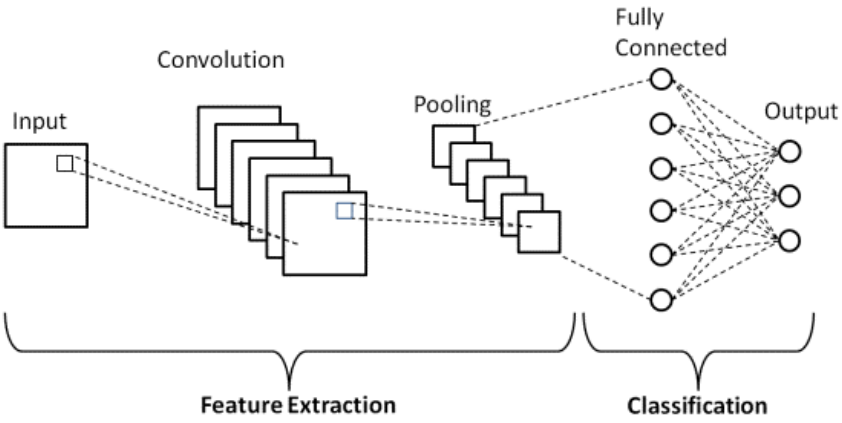


Figure 2.3: CNN basic structure

2.3.3 Privacy concerns in camera-based system

In order to function, computer vision requires many images and therefore acquisition systems that are sufficiently fast and have good temporal and spatial resolution.
Among these, RGB cameras have been widely used for their ability to acquire detailed color images, which allow an accurate representation of scenes, thus facilitating the recognition of objects, faces and characteristics of the environment.
The large amount of visual details captured in the image also makes it easier to train ML models.
However, this comes at a high cost, especially considering the applications and contexts in which these systems operate: privacy and security.
Since RGB images include sensitive data, such as people's faces and physical features, there is a possibility that specific subjects may be identified, especially in public spaces or in personal monitoring applications.
This is particularly relevant if the data is transferred or stored in cloud systems, where it may be exposed to risks of unauthorized access, violations, or used for unforeseen purposes.
The risk is also accentuated by the possibility of associating images with location data and the acquisition of information regarding patients' home environments, including any relatives or family members of the same.
To address this problem, sensors have been developed that are able to capture only the information considered necessary, thus avoiding capturing the surrounding environment or details that are not essential to the analysis, such as colors or features.
Examples of these sensors are the *event cameras* and the *depth sensors*, both used in this project will be discussed in more detail in the following chapters.

Chapter 3

Neuromorphic Systems

3.1 Neuromorphic Engineering

Neuromorphic engineering is an interdisciplinary subject that takes inspiration from biology, physics, mathematics, computer science and electronic engineering that aims to design artificial neural systems whose architecture and operation are based on those of biological nervous systems[7].
Although it is a subject of study and research today, its invention dates back many years and a first application of its principles was proposed by Carver Mead in 1980. Mead proposed using the principles of biology to design electronic circuits that mimic neuronal functions. His goal was to overcome the limitations of traditional computing architectures, improving energy efficiency and real-time processing capabilities.
In traditional computing systems, in fact, information processing and storage occur in separate areas (Von Neumann architecture) thus leading to a continuous transfer of data between these two entities that is inefficient in computational and energy terms, causing a lot of bottleneck issues.
Especially nowadays with the widespread diffusion and adoption of AI and its applications across all sectors, the issue of energy consumption is one of the main problems to be addressed.
With traditional computers training neural network models, or performing complex tasks on large amounts of data, is becoming increasingly energy-intensive and these are often not very flexible in adapting to changing conditions. A human brain is estimated to use just 20 watts to perform complex tasks, unlike these power hungry computers that require much more for the same operation.
In this context neuromorphic systems try to solve these problems through parallel and asynchronous processing that emulates the brain's neural networks.
At the base of these systems are neurons, or rather artificial versions of biological

neurons. These are the fundamental units of the human brain responsible for processing and transmitting signals through electrical impulses called "spikes".
In an attempt to emulate the behavior of the human brain, two different approaches have been explored, software side and hardware side, that will be introduced in the following paragraphs.

3.1.1 Neuromorphic Sensing

Neuromorphic sensing is an approach to data acquisition and processing inspired by the functioning of the biological sensory system, where sensors mimic the behavior of neurons in capturing and transmitting information efficiently [7].
Unlike traditional sensors that generate a continuous stream of data at a constant sampling rate, neuromorphic sensors produce asynchronous events that are generated only in response to significant changes in the environment.
The asynchronous nature of these sensors, based on the acquisition and processing of events, significantly reduces the amount of data to be processed, thus improving their energy efficiency and real-time capability.
This new approach makes neuromorphic sensing particularly useful and suitable in sectors such as robotics, augmented reality and gesture recognition applications as in our case.

A prime example of a neuromorphic device is the event camera, commonly referred to as a dynamic vision sensor. These devices excel at capturing "events," which correspond to changes in brightness at individual pixels as they occur.
This new type of camera was developed taking inspiration from the human biological eye that is able to capture only the information deemed useful in a highly responsive and low-energy way [8].
Tactile, acoustic, and olfactory devices have also been developed, all designed to mimic the biological processing of sensory information, but will not be explored further as they are not relevant to this project.

Although their use is not yet widespread in common applications, these sensors offer numerous advantages over traditional sensors including:

- *Reduction of energy consumption*: as explained above, there is less data to process but more relevant in terms of importance of the information acquired, leading to improved energy efficiency.
- *Low latency*: these sensors are extremely reactive to environmental changes, being able to acquire information on the order of microseconds.
- *Ability to operate in dynamic environments*: they can be used in extreme environmental conditions, such as too little or too much light, in motion, etc.

In the following chapters the event camera sensor will be explained and explored in more detail, as it was directly used in the creation of the dataset.

3.1.2 Neuromorphic Software

The problem of excessive energy consumption by artificial neural networks has led to the development of new network models, called Spiking Neural Networks [7].
As in normal artificial neural networks, SNNs are made up of several interconnected layers, each of which is composed of neurons.
Spiking neurons, as the name suggests, receive and emit "spikes", binary signals very similar to those exchanged in our brain, replacing the floating point values exchanged in the most common ANNs.
The advantage of having discrete values, 1 or 0, as signals removes the computational complexity due to the multiplications between weights and high-precision activations values, typical of ANNs, that are responsible for high consumption and also delays.
Instead of having floating point values that encode information based on intensity, SNNs work with single-bit binary activations that encode information over time.
The concept of time is very important and represents a fundamental difference from normal models.
Spike neurons operate on a weighted sum of inputs, spikes trains of other neurons, which are generated over time just when certain conditions are met. Rather than passing the result to a sigmoid or relu function, as in ANNs, this weighted sum contributes to the time-dependent membrane potential U(t) of the neuron itself.
If sufficiently stimulated the membrane potential of the spiking neuron exceeds the potential threshold and, when this condition is reached, the neuron emits an output spike that will be transmitted as input to all the other neurons connected to it.
On the contrary, if the membrane potential does not reach the threshold, its value will tend to decrease over time, until it returns to a stable state. This effect is called "Leakage effect".
The neuron model just described is called "Leaky Integrate and Fire" neuron, abbreviated LIF, and emulates well the real behavior of biological neurons.

In the following chapters SNNs will be further explored along with the possible learning methods available, with more attention to the design choices used in this work.

3.1.3 Neuromorphic Hardware

To make the most of the advantages and features of neuromorphic software, hardware architectures have been developed over the years that draw direct inspiration from the physical structure of the biological brain and its functioning [9].
The structure and functioning of these devices are based on the neuron, the basic computational unit, and the synapses, connections between neurons, which allow the transmission of information.
The information is encoded through spikes, which are events generated by neurons asynchronously, that is, only when certain conditions occur.
A fundamental difference compared to the classic architectures commonly used lies in the location of processing and memory.
In Von Neumann architectures, in fact, the data is processed by a CPU that recalls them from a memory and stores them in the same memory at the end of the processing.
The limitation of these architectures is that very often the CPU and memory are on different chips and the transfer of information between the two parts is energy-intensive and can cause bottlenecks that reduce speed.
Neuromorphic devices have been developed precisely to solve these problems. As already explained above, these neuromorphic chips are made up of many neurons that communicate with each other via synapses. The novelty lies in the fact that the processing and storage of information takes place in the same place, inside the neuron, avoiding the normal exchanges between memory and computational units of normal computers.
In addition to improving performance and reducing consumption, this approach introduces other advantages.
Each neuron, being a computational unit in itself and having its own independent memory, can work in parallel with the others without dependencies of any kind.
Furthermore, their distributed nature allows easy scalability and high fault tolerance.
If you want to increase the network, you will simply need to add new chips to the existing ones. Likewise, if a particular chip fails, the system can still function properly because the information is distributed across many places.
Because of these characteristics, these chips are well suited to running machine learning algorithms at the edge, where there are physical limits of space and energy.
Other potential applications of this technology include self-driving cars, drones, robots, smart home devices, and voice and image processing.
Furthermore, creating systems that emulate the human brain could help us better understand how they actually work and thus create systems that are ever closer to their biological counterparts in nature.

Some examples of neuromorphic hardware are:

- *IBM's TrueNorth chip*: a neuromorphic chip developed by IBM that simulates one million neurons and 256 million synapses. It uses a spiking neuron network and an event-based structure, enabling highly parallel processing with very low energy consumption. It is 10,000 times more energy-efficient than conventional microprocessors and only uses power when necessary.
- *Intel Lab's Loihi 2*: has two million synapses and over one million neurons per chip. They are optimized for SNNs and support real-time learning through synaptic plasticity mechanisms such as Spike-Timing-Dependent Plasticity (STDP).
- *SpiNNaker*: a neuromorphic supercomputer developed by the University of Manchester, designed to simulate large-scale neural networks. Each SpiNNaker chip contains 18 processing cores capable of simulating neurons in real time. It has been used to simulate detailed models of the brain, such as the mammalian cerebral cortex.

3.2 Event Cameras

Event cameras are the most representative example of a neuromorphic sensor.
Also called "Dynamic Vision Sensors", their development takes inspiration directly from the human retina and its particular sensitivity in detecting rapid changes in the scene and in leaving out everything that is immobile and devoid of useful information.
The main difference compared to normal visual acquisition systems, such as RGB cameras, concerns precisely this characteristic: the ability to record useful information only when it occurs.
Traditional cameras record videos in the form of sequences composed of different frames, or snapshots of the current scene, captured at regular intervals with a fixed frequency defined in frames per second (fps).
A *synchronous* approach of this type has limitations in terms of redundancy of the information acquired, with consequent high flow of data generated, and energy consumption that can be expensive especially in static scene situations.
In event cameras each pixel, composing the sensor, is independent from the others and is able to generate information when the pixel itself detects a change in brightness in the scene, in a completely *asynchronous* manner. The information generated by the individual pixels is called "*event*" and is encapsulated in a packet containing:

- the coordinates (X,Y): spatial coordinates of the pixel within the matrix.
- the polarity: can be positive or negative, indicates the direction of the detected change in brightness.
- the timestamp: instant in time in which the detection occurs.

These sensors record the scene in the form of a series of events, which are generated only if changes in brightness have occurred at the level of individual pixels. The variations in brightness signal that the subject is moving or the scene is changing, situations in which the camera must record.
In addition to energy efficiency and the reduction of redundant data, event cameras have two major advantages: a *high temporal resolution* and a *high dynamic range*. They are capable of recording scene changes in the order of microseconds, which makes them ideal for applications that require high speed and temporal precision, avoiding unwanted phenomena such as blurring.
Furthermore, having a high dynamic range, they are able to operate in low-light and high-contrast scene situations, since they detect changes rather than static scenes, offering greater adaptability to different lighting conditions.
In Fig.3.1 a comparison is shown between the two outputs acquired respectively by a frame based camera and an event camera.

The subject is a rotating disk of uniform color with a black dot drawn on it whose movement is to be traced over time. The frame camera captures frames at fixed time intervals, including the rotating disk in the images, irrelevant and redundant information, but failing to accurately track the trajectory of the black dot, which appears to jump in time due to the lack of intermediate frames.
Looking at the output of the neuromorphic sensor, can be observed how the dot is accurately tracked in time, represented as a continuous series of events, while the information regarding the disk is completely absent, as it is perceived as static and therefore irrelevant for the purposes of capture.
The above-mentioned features make event cameras particularly suitable for appli-

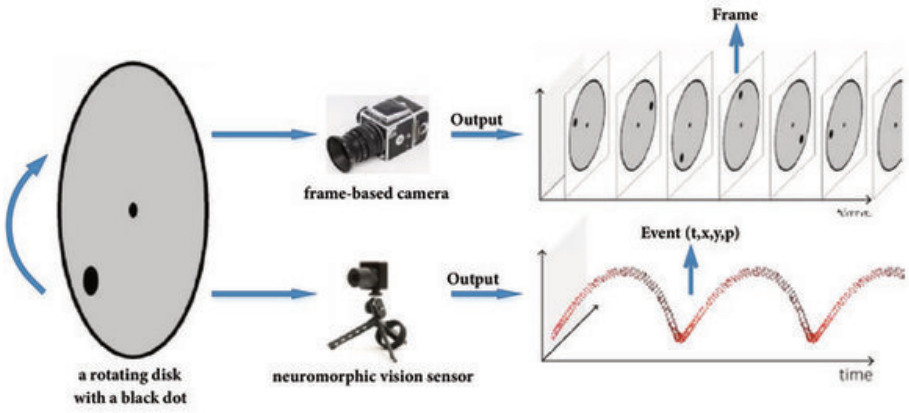


Figure 3.1: Frame vs Event based Cameras

cations that require a rapid response and at the same time a detailed and accurate representation of movement.
Robotics, drones and autonomous driving benefit from them, allowing real-time detection of obstacles in extremely variable lighting conditions.
The high temporal resolution and the ability to filter movement from other sensitive information, that is not of interest, make these sensors very suitable also for diagnostic and monitoring applications in which it is of primary interest to guarantee privacy and security of the information acquired regarding the patient.

Feature	Frame based camera	Event Camera
Temporal Resolution	33 ms (30 fps)	1 μs
Spatial Resolution	Up to 12 MP	Around 0.3 - 1.2 MP
Energy Consumption	1-10 W	10-100 mW
Dynamic Range	60-70 dB	120-140 dB
Pixel Bandwidth	Constant, up to 1 Gbps	Variable, 10-100 Mbps

Table 3.1: Frame based vs Event Camera

3.3 Spiking Neural Network

Spiking neural networks are part of what are called neuromorphic algorithms, that is, artificial networks and models that simulate the principles of brain functioning, capable of working and processing information through *spikes*.
Unlike floating point values, commonly used by neurons in ANNs, spikes are single-bit binary activations that encode information in time rather than intensity and represent the most plausible explanation to date of how the brain is able to process and exchange information.

3.3.1 LIF model

SNNs are formed by groups of interconnected neurons organized in layers, having inputs and an output that are encoded in spike trains. In Fig.3.3 a schematic model of a spiking neuron is shown.
There are many mathematical models able to describe the temporal evolution of the membrane potential of the biological neuron, among these the most used is the Leaky Integrate and Fire model, abbreviated "LIF".
The first to hypothesize a model of this type was Lapicque in 1907 [10] who tried to stimulate the nerve fiber of a frog's leg using a current source and observing how long it took to contract based on the amplitude and duration of the current.
From his experiment he concluded that the behavior of a spike neuron was similar to a low-pass filter circuit RC, composed precisely of a resistance R and a capacitance C.
In Fig.3.2 [7] can be observed in (a) the bilipid membrane of a neuron that separates the intracellular medium from the extracellular medium, which is represented in (b) as the capacitance C of the circuit, and an ion channel that allows charge carriers to diffuse through the membrane by opening and closing. The latter is represented in (b) as the resistance of the circuit.
The membrane dynamics, modeled using an RC circuit, can be represented as:

$$\tau \frac{dU(t)}{dt} = -U(t) + I_{\text{in}}(t)R \tag{1}$$

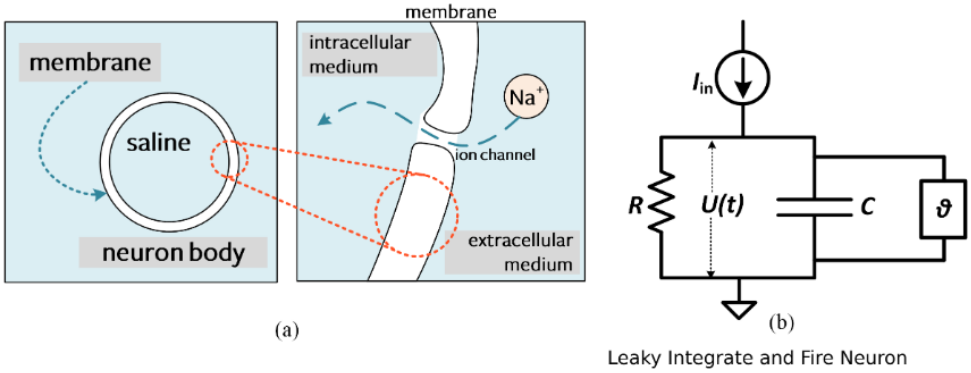


Figure 3.2: Biological membrane(a). RC membrane model(b)[7]

By solving (1), considering the case of constant current and making a series of approximations not reported here, we obtain the expression of the membrane potential:

$$U[t] = \underbrace{\beta U[t-1]}_{\text{decay}} + \underbrace{WX[t]}_{\text{input}} - \underbrace{S_{\text{out}}[t-1]\theta}_{\text{reset}}. \tag{2}$$

This expression represents the membrane potential of a neuron that varies over time based on the input, the current membrane potential and a possible reset due to the activation of the neuron at the previous step.
As can be observed from (2) 3 terms can be identified:

- *Decay*: represents the membrane decay with respect to the value at the previous step *U[t-1]*. The parameter β is defined as "membrane decay factor", it takes values between 0 and 1 and indicates how quickly the potential dissipates over time. It represents the "Leaky" behavior of the LIF model.

- *Input*: *WX[t]* represents the current input arriving at the neuron. W represents the synaptic weight associated with *X[t]* which represents the value of the input at time *t*. For simplicity both *X[t]* and W refer to the weight of a single input to a single neuron. Normally the input is vectorized and the weights are represented using a matrix.

- *Reset*: this term represents the reset of the membrane potential that occurs when the neuron has generated a spike at the previous step. In fact, if the membrane potential exceeds a certain threshold defined as θ, the neuron activates and at the next time step it will subtract its threshold from the current membrane potential, so that the neuron returns to the inactive state and does not emit unnecessary spikes. This term has no effect if $S_{\text{out}}[t-1] = 0$.

If it happens that $U[t] > \theta$ at time t, the neuron will be activated and generate a spike:

$$S_{\text{out}}[t] = \begin{cases} 1, & \text{if } U[t] > \theta \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Otherwise the membrane potential will be updated based on the current inputs and weights and the neuron output will have value 0.
The behavior described above is summarized in Fig.3.3.

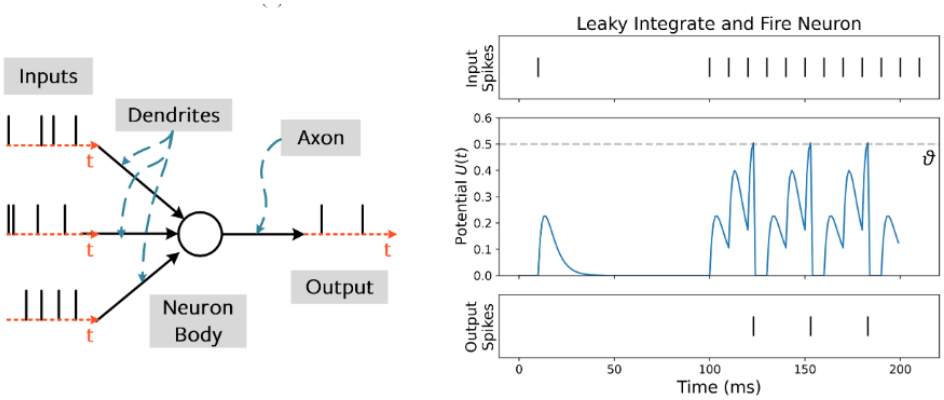


Figure 3.3: Behavior of a LIF neuron [7]

3.3.2 Spikes Encoding

SNNs, like the human brain, acquire information from the outside world and forward it to neurons in the form of spikes.
In order for these spikes to carry information, they must be encoded appropriately. It is therefore necessary to introduce the concepts of:

- *Input encoding*: conversion of data, in input to the network, into meaningful spikes.
- *Output decoding*: training the network to emit output in the form of spikes that are meaningful and informative.

In our experiment, the input data to the network were generated by an event camera, a neuromorphic sensor that, as already introduced, produces events having the same nature as spikes since they are generated asynchronously when a change in brightness is detected.
For these reasons, *input encoding* is not necessary, however, for completeness, the most commonly used input encoding methods will be introduced:

- *Rate coding*: in this approach, the intensity of the information to be converted is translated into the frequency of the spikes. The greater the intensity, the greater the frequency with which the spikes are launched. This encoding is very similar to the functioning of the human retina: the photoreceptor cells absorb the light and translate it into spikes. A greater brightness corresponds to a high firing rate.

- *Temporal coding*: also known as "latency coding", in this encoding the number of spikes in time is no longer relevant, but what matters is *when* a spike is generated. In this approach, a particularly intense information (such as a high brightness) is encoded through a single spike generated immediately, with a low latency, hence the name. On the contrary, a dark input will be encoded with a spike generated with a certain delay or not generated at all. This type of encoding gives much more importance to the individual spikes.

- *Delta modulation*: this type of encoding is very similar to the operation of event cameras since it focuses on variations rather than absolute values of the signal. Every time the signal varies and exceeds a certain threshold, a spike is generated.

The same coding techniques presented above can be used to decode the output generated by the network in a meaningful and interpretable way.
Taking as an example a classification problem, with N possible classes, and applying an input to the network, how can the spikes generated by the neurons of the output layer be interpreted?
Using a *rate coding* policy the predicted class will be represented by the neuron with the highest firing rate that therefore presents the highest number of spikes generated.
On the contrary, using a temporal approach such as *latency coding* the predicted class will be that relating to the neuron that fired first.
Another technique used for output decoding is called *membrane potential decoding*, in which the output is determined by analyzing the membrane potential of the output neurons, rather than counting or timing the spikes.
The choice of the encoding method that best suits your needs depends on many factors such as the type of data that the network receives as input, the noise that characterizes this data, the computational resources available and the robustness of the model that you want to obtain.
For example, a rate code policy promotes robustness by creating a model with a higher error tolerance, since if a neuron fails to fire at a certain time, other spikes will still be generated. Furthermore, having more spikes also implies that the model will learn more, especially when using error propagation-based learning methods as we will see later. However, this entails limitations in terms of energy consumption

and speed, which would not be the case when adopting a time-based approach with fewer spikes, such as *latency code*.

In order to learn, SNNs need an objective function that measures how far the predicted result is from the desired target. These are also called "loss functions" and vary based on the decoding that has been chosen.
Among the most common and used objective functions are:

- cross entropy loss
- mean squared error (MSE)

Taking into consideration the case of *rate decoding*, the *cross-entropy loss* is applied to the number of spikes produced by the neurons in the output. The goal is for the neuron associated with the correct class to emit the greatest number of spikes, while those of the wrong classes produce few or none. A very similar approach is applied to membrane potentials, encouraging the membrane potential of the correct class to increase and that of the neurons representing the wrong classes to decrease. Instead, considering the mean squared error, a target spike rate is assigned to each class by specifying the expected number of spikes for each. For example, the correct class is set to fire during 80% of the total timestamps, while the remaining classes are expected to fire during the remaining 20%. This approach is effective in keeping neurons active and close to the firing threshold.
The final goal is to minimize and use these functions so that the network improves in the task to which it has been assigned.

3.3.3 SNN's learning approches

There are two main learning techniques that operate with very different principles and approaches: *BPTT* and *STDP*.

Backpropagation Through Time, abbreviated BPTT, is a supervised algorithm that allows the network to learn by exploiting the loss function calculated for each prediction during training. In fact, at each iteration, the network has the desired target and is able to calculate how far the current prediction is from the correct one.
Once this error is obtained, the network is unrolled in time and its gradient is calculated for each time step. The gradient, whose expression is in (4), is a vector of partial derivatives of the error L with respect to each weight w_{ij} of the network and indicates how much a given weight contributes to the error and, consequently, how this must be modified to reduce the final loss function. As the name itself suggests, the gradient is then back propagated by updating the weights in order to

reduce the error along all the steps.

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial S_{\text{out}}} \cdot \frac{\partial S_{\text{out}}}{\partial U} \cdot \frac{\partial U}{\partial w_{ij}} \tag{4}$$

However, in order for the technique described above to be used in SNNs, it is necessary to make a consideration: spikes, as discrete impulses, are not differentiable so the term $\frac{\partial S_{\text{out}}}{\partial U}$ in (4) is 0.
It is therefore necessary to introduce the concept of *surrogate gradient*, i.e. a gradient of a *continuous* surrogate function, that approximates the behavior of the spikes of a given neuron. In this way, the network can use a continuous, differentiable signal to calculate the gradient.
Common surrogate functions include:

- *Sigmoid Function*:
$$\sigma(z) = \frac{1}{1 + e^{-\beta(z-\theta)}}$$
where β is a slope parameter and θ is the threshold. It is a continuous function that approximates the behavior of the spike around the threshold.

- *Hyperbolic Tangent*:
$$\sigma(z) = \tanh(\beta(z - \theta))$$
Provides a value between -1 and 1, and allows for a more gradual variation of the gradient.

- *Gaussian Function*:
$$\sigma(z) = e^{-\alpha(z-\theta)^2}$$
It is centered on the threshold θ and controlled by a parameter α that adjusts the width of the bell. It works well for more localized gradients around the threshold.

The other learning approach bases its operation on the most plausible mechanism to date with which the human brain learns, namely *synaptic plasticity*.
Spike-Timing-Dependent Plasticity, abbreviated STDP, is an unsupervised learning technique in which the relative timing of spikes between pre and post-synaptic neurons plays a major role.
More specifically, if a pre-synaptic neuron sends a spike immediately before the post-synaptic neuron, the synaptic weight tends to strengthen (synaptic potentiation); vice versa, if the post-synaptic neuron fires before the pre-synaptic neuron, the connection weakens (synaptic depression).
Following this mechanism, the network is able to learn temporal patterns without having an objective function to minimize and without the need to specify the target as input. However, since this technique is not suitable for classification problems, it was not used in our network.

3.4 Combining Event Camera and SNN

As already introduced in the previous paragraphs, the asynchronous and temporal nature of SNNs, based on spikes, combines perfectly with the data produced by neuromorphic sensors such as event cameras.
The reason behind this compatibility is due to the fact that the events themselves, generated by these sensors, are asynchronous and do not contain redundant information unlike continuous frames.
The combined use of neuromorphic sensors and SNNs therefore allows to drastically reduce the amount of data to be processed, improving processing speed and energy efficiency. Unlike ANNs, where each neuron calculates a numerical value at each time step, in SNNs neurons activate and fire only when the membrane potential reaches a certain threshold, thus reducing the number of calculations performed.
Furthermore, the binary nature of the individual spikes simplifies and speeds up the multiplications between the weights and the input signals to the neurons.
The acquisition speed and energy efficiency of these sensors, combined with the computational efficiency of SNNs, makes these systems particularly suitable for real-time applications, such as robotics and autonomous driving systems, where it is essential to acquire and process information as quickly as possible with low response latency.
Edge applications where computing power and resource availability are limited also benefit from this, thanks to the reduced amount of information acquired and the lower energy consumption to process it.
Despite these technologies being promising thanks to their intrinsic advantages, there are still too few neuromorphic datasets in circulation and they are often limited to controlled scenarios and not representative of complex real applications.
In the next paragraph, some of the most used ones for training and evaluating SNNs will be introduced, from which we took inspiration for the creation of our dataset.

3.5 Existing Neuromorphic Datasets

Although many hand gesture datasets have been created in recent years, most of them have been realized using common sensors such as standard RGB color cameras.
The importance of neuromorphic datasets to advance research in event-based computer vision has prompted Hu et al. [11] to fill this gap by converting four frame-based datasets into event-based representations using a DVS camera pointed at a screen displaying frame-based images intermittently.
A similar approach to the one described above involves converting frame-based

datasets while keeping the image *static* and moving the neuromorphic sensor. This method is biologically more realistic as it simulates the saccadic movements of the human eye, performed to acquire information, reducing the loss of temporal resolution and the introduction of unwanted artifacts caused by the monitor refresh. With this technique the *N-MNIST* was created, the neuromorphic version of the more famous MNIST, a dataset representing images of handwritten digits.
The alternative to the conversion of frame-based datasets, already created, is the creation from scratch using neuromorphic sensors directly. This is what Amir et al. did in [12] for the creation of *DVSGesture*, a pure neuromorphic hand gesture dataset created using the DVS128 camera.
In the following paragraphs a brief introduction of the above-mentioned datasets will be provided.

3.5.1 N-MNIST

The N-MNIST is a spiking version of the traditional frame-based MNIST dataset. It contains the same 60,000 training examples and 10,000 test examples as the original MNIST and is kept at the same visual scale (28x28 pixels).
It was created using an ATIS sensor mounted on a motorized pan-tilt unit, which moved while observing MNIST examples displayed on an LCD screen.

3.5.2 DVS Gesture

The DVS Gesture Dataset is designed for the recognition of human gestures captured with a DVS128 event camera.
It includes 11 categories of gestures performed by 29 subjects in different lighting conditions.

Chapter 4

Dataset Creation

As discussed in the previous chapters, the scarcity of neuromorphic datasets limits the research, development and diffusion of event-based computer vision applications. The aim of this thesis is to fill this gap, creating a large scale neuromorphic hand gestures dataset in order to make it available for future research in promising fields such as telemedicine or more generally in applications based on human computer interaction.
In this chapter, the dataset creation process will be presented. We will start by describing the cameras used, the setup and their synchronization, and then we will delve into the structure of the experiment with a particular focus on the selected tasks.
The acquisition process will also be illustrated in detail, including the various steps to collect the data while maintaining the synchronization between the various devices. Finally the numerical characteristics of the dataset obtained and its organization will be presented.

4.1 Multi-camera Setup

The main objective of this work concerns the creation of a large neuromorphic dataset, obtained through the use of a particular camera, introduced in chapter 3, known as *neuromorphic sensor*.
To make the dataset in question comparable with other traditional acquisition forms, it was decided to combine the neuromorphic sensor with two devices capable of recording depth information and RGB frames, together with a marker-based system commonly used as a reference in motion tracking applications.
The entire process of acquisition of the tasks was held in a limited access laboratory granted by the *National Research Council* (CNR) and the setup of the cameras remained unchanged for the entire duration of the collection, in order to guarantee

uniformity of conditions throughout the acquisition process.
The sensors used in the experiment, which will be introduced briefly in the following paragraphs, were:

- 1 DVXplorer Lite (neuromorphic sensor)
- 2 Kinect Azure (depth sensor)
- 1 Optitrack (motion capture system)

The cameras were arranged as shown in Fig.4.1, with the DVXplorer in the center and the two Kinects on the side at equal distance from the event camera. The chair

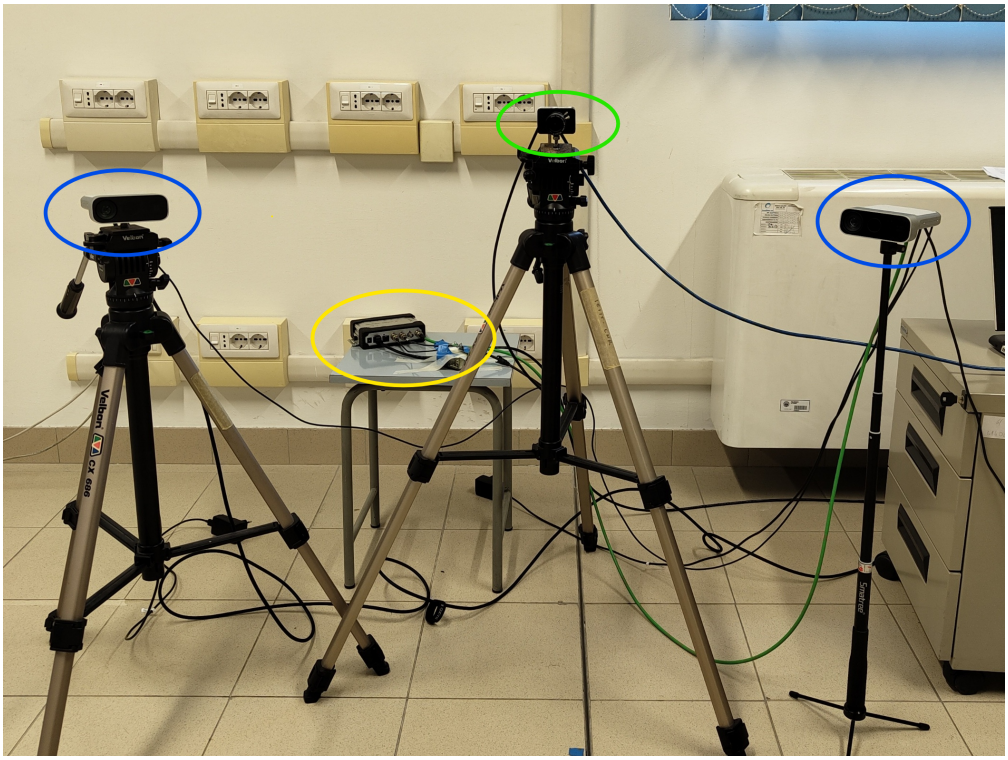


Figure 4.1: Multicamera setup: DVXplorer Lite in the green circle, Azure Kinect in the blue ones. Esync in yellow for camera synchronization

used by the subject during the tasks was positioned in front of the event camera. The distance between the camera and the chair was always fixed with adjustments in the order of +/- 10 cm based on the length of the subject's arm, in order to maintain the movement of the hand central to the field of view of the cameras.
Everything was designed to have the subject in the center of the room so that he was equidistant from the six cameras fixed to the walls that make up the optitrack

system.
In Fig.4.2 the participant's chair is shown in the orange box, 3 of the 6 cameras that make up the optitrack in the pink circles and the two operator stations indicated by the blue arrows.
The windows were darkened to maintain the same lighting values between various

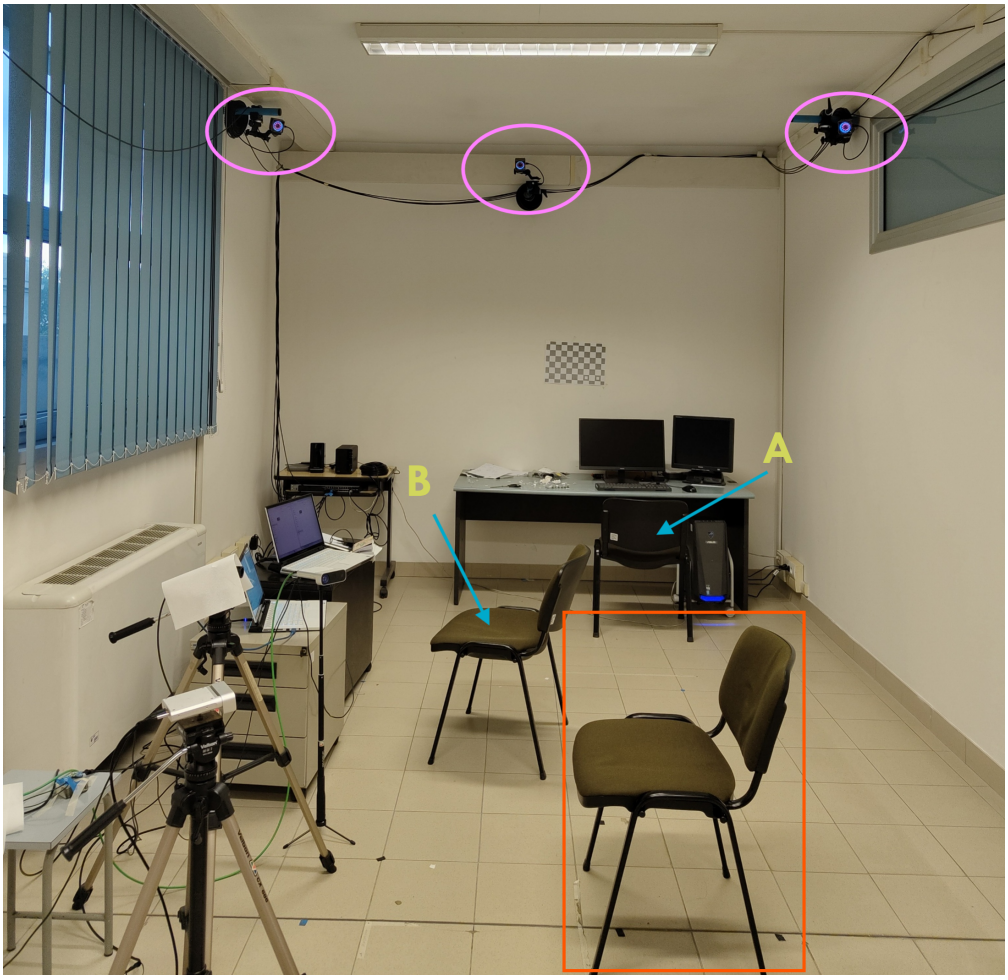


Figure 4.2: Lab setup: subject position in orange, optritrack cameras in pink and operator positions indicated by blue arrows A and B.

recordings and any reflective surface was covered so that it was not perceived as a marker by the motion capture system.
The kinects and the neuromorphic camera were connected to the computer managed by operator B who controlled their correct functioning, while operator A had control over the optitrack and the esycn shown in figure Fig.4.1.
The esync, a hardware device directly connected to the optitrack system, was in

turn connected to the kinects and the event camera via BNC connectors so that it acted as a master clock and sent them signals for correct synchronization.

4.1.1 DVXplorer Lite

The DVXplorer Lite was the main sensor of our experiment. It is a neuromorphic camera developed by a Swiss company, Inivation, specialized in the production of event cameras.
The sensor has a spatial resolution of 320x240 pixels, a temporal resolution of 200 µs and a dynamic range >90dB.
These characteristics give it an extremely low latency (in the order of microseconds) and at the same time allow it to operate effectively in variable lighting conditions.
The neuromorphic event nature, based on the detection of variations in light intensity rather than on a constant acquisition of frames, allows it to reduce energy consumption. In fact, the average energy consumption is around 140/150 mW during normal operation.
The sensor can be interfaced with a PC from which it can be configured and controlled through a proprietary software called "DV". The software is equipped with a graphical interface and allows you to visualize, record and play back data.
It is possible to define customized projects based on needs, characterized by a structure formed by logical blocks called "DV modules". These modules can be added, removed and connected to each other, allowing you to aggregate, filter and convert the stream of events, acquired by the camera, based on needs.
The files are in a custom format called AEDAT (Address Event Data) designed to store the event data and its timestamp information:

$$(timestamps, polarity, x, y)$$

The advantages of this format include compactness, time precision and high efficiency of the generated files which can then be processed easily and precisely in real-time applications and embedded systems.

4.1.2 Kinect Azure

The Kinect Azure, developed by Microsoft, is a multifunctional camera designed for depth tracking, motion capture, and high-resolution RGB image acquisition [13].
It uses Time-of-Flight (ToF) technology to accurately measure distances, providing a depth resolution of 1024x1024 pixels with a range of 0.5 to 4 meters.
This technology is able to estimate the distance between the camera and the objects in the frame by calculating the time it takes for a light pulse to travel the

camera-object-camera path.
The RGB sensor offers a resolution of up to 3840x2160 pixels, allowing for detailed scene acquisition.
The device also includes a microphone array (not used in this context) to capture spatial audio and has a wide field of view (FoV), equal to 120°x120° for the depth camera.
For the purposes of the experiment, it was decided to use two Kinects triangulated with the subject, in order to capture in detail every facet of the movement performed. Both cameras were connected to a PC via USB connector and driven by two different bash scripts that, once executed, made them wait for the signal coming from the master (the OptiTrack).

4.1.3 OptiTrack

The OptiTrack system is a motion capture platform designed for precise motion tracking applications.
It is composed of a network of high-speed infrared cameras capable of detecting the position of markers in space with submillimetric precision.
The markers used in this experiment are small spherical objects with an adhesive base defined as *passive*, meaning they do not emit their own light but are coated with a highly reflective material that reflects the infrared light emitted by the system's cameras.
The operation can be divided into 3 phases:

1. Emission and detection: once a recording has started, each camera emits infrared rays and captures the visible markers in its field of view as light dots.

2. Triangulation: this process allows the position of the markers in space to be detected using the 2D coordinates of a given marker detected by at least two cameras. These coordinates are processed by the system to calculate the precise 3D position of the marker in space.

3. Assignment: the Motive software maps the markers on a predefined model (in our case a human skeleton) to track the movements performed by the subject.

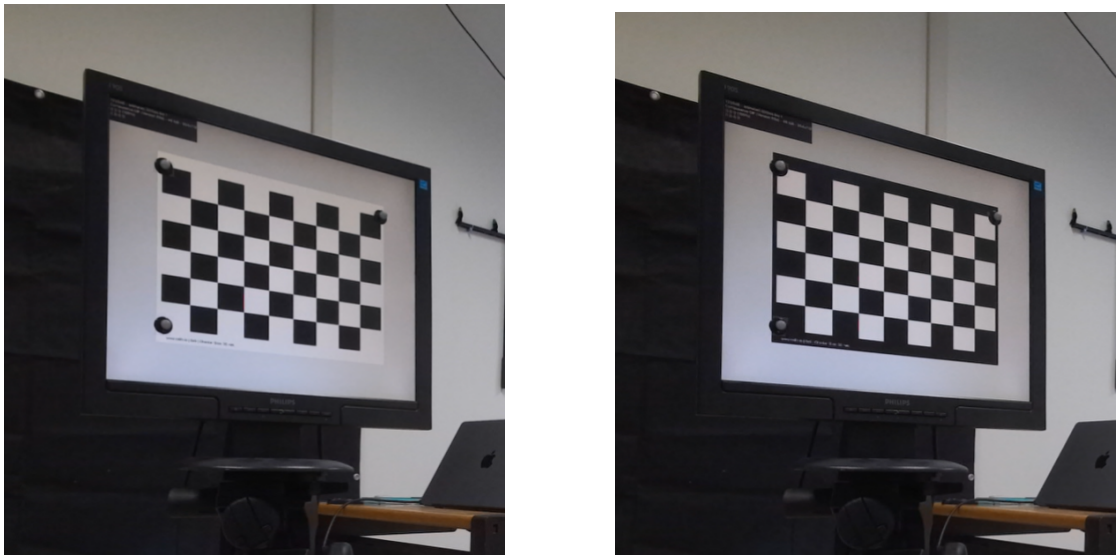
Due to its high precision, this system is often used as ground truth in scientific research, especially to evaluate the performance of other devices.
The system is also equipped with a hardware module, the eSync, which allows precise temporal synchronization between different cameras, making it perfect for multimodal experiments.

4.1.4 Camera Calibration

In a multimodal context like ours, consisting of several sensors, the camera calibration process is very important as it allows to align the devices by synchronizing their spatial coordinates.
It consists in performing a series of recordings by placing a checkerboard, a simple grid printed on paper, in front of the cameras at different distances and angles.
The frames relating to the same test, acquired by different cameras, will then be aligned using specific software.
In this section, only the acquisition process of these tests will be described, while the mapping of the data in the same global coordinate system was not part of this work.
For the Kinect and OptiTrack it was decided to reuse the calibration scripts, as they had already been implemented during other projects.
For the DVXplorer it was necessary to create a new project within the DV software, cascading the *Dvxplorer*, *Accumulator* and *Calibration* modules.
As a checkerboard a 6x9 with a square size of 30 mm was used, which was printed on a sheet and shaken back and forth during the acquisition, so that the events were generated.
However, it was found that the DV software was not able to align the checkerboard patterns precisely enough and, by analyzing the intrinsic parameters generated within the calibration file, it was found that the calibration error was too high (a parameter that indicates the quality of the calibration).
It was therefore decided to create a GIF, alternating two images of chessboards with inverted black and white colors, and showing it to the cameras through a monitor positioned on a pedestal Fig.4.3.
In this way the DVXplorer was able to generate events more precisely and the accumulated frames were sharper, especially the edges of the squares .
The calibration process was repeated correctly for 40 poses, varying the distance and angle of the monitor with respect to the cameras.

4.1.5 Camera Synchronization

Creating a multimodal dataset using different types of sensors involves prioritizing the synchronization of devices with each other and ensuring that recordings start at the same instants in time or with at most a small known delta.
In the synchronization process, OptiTrack was considered as the master because, as previously mentioned, it was designed and studied specifically for use in multimodal contexts.
The eSync, integrated into the system, allows it to act as a master clock and synchronize external devices by distributing reference signals.
The kinects in turn were previously synchronized during past research work and



(a) Checkerboard 1 **(b)** Checkerboard 2

Figure 4.3: GIF Checkerboard, inverting colours.

configured so that, once the related scripts were launched, they went into a waiting state and started recording only when a recording was started by the master (OptiTrack).
The synchronization of the DVXplorer was instead part of this work and the process will be described below.
As can be seen in Fig.4.4, the event camera has an extra *SYNC_IN* port that allows it to interface with third-party cameras.
This port can be configured, through the DV software, to detect rising edges of signals from external devices or cameras. When this happens, a *trigger event* of type *EXTERNAL_SIGNAL_RISING_EDGE* is generated.

The eSync was then connected via an HR10A cable to the SYNC_IN port and a recording was started, by the DVXplorer and by the OptiTrack simultaneously.
The signal generated by the eSync is a train of 30Hz pulses, whose rising edge represents the instant of the OptiTrack acquisition start.
The file generated by the event camera, in *aedat4* format, was then analyzed via a python script. However, by analyzing the individual packets of the file, it was found that too few triggers had been fired and it was assumed that the few detected were spurious and due to other causes.
By connecting the device to an oscilloscope we verified that the 30 Hz signal was actually generated and it was.
It was then discovered that the detection circuit, present inside the DVXplorer, is

Figure 4.4: DVXplorer sync connectors

electrically isolated from the rest of the camera and, in order for the camera to detect signals, coming from external cameras, it is necessary that *Vdd* is supplied to the circuit itself.
Vdd was therefore supplied with the same voltage as the SIGNAL_IN signal. Since this voltage comes from an external source, the grounds were connected together so as to have a common ground Fig.4.5.
The synchronization process was then repeated and the aedat4 file was generated and analyzed again: the DVXplorer and the OptiTrack were correctly synchronized.

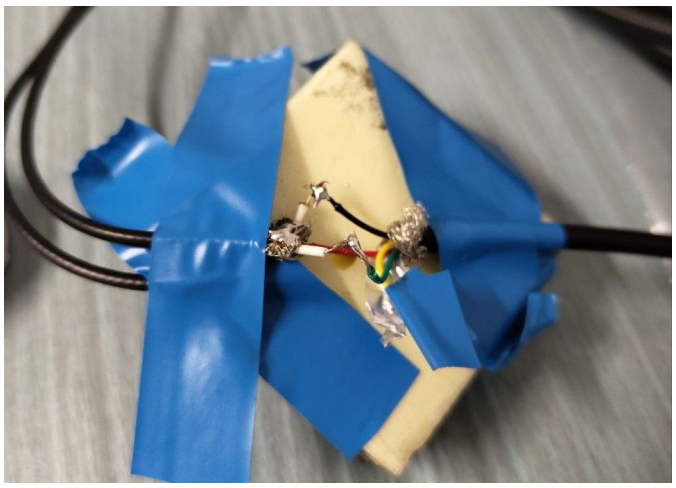


Figure 4.5: Different GNDs were welded to have the same reference

4.2 Experiment Protocol

In this section, the protocol used to create the dataset will be explained in detail and the tasks performed by the subjects during the experiment will be illustrated. The experiment involved 25 healthy people, recruited from the student population, whose hands were measured in order to collect useful information during the splitting phase of the dataset or for future research.
Each participant was asked to remove clothing or reflective objects so that they did not interfere with the detection of the markers. The subject was then made to sit at the station in front of cameras and the markers were applied to the hand.
The positioning of the markers varied based on the task to be performed, and the order of the tasks was designed to minimize the movements of the markers. The Fig.4.6 illustrates the arrangement of the markers relative to each task.
The tasks analyzed were 5:

- *Finger Tapping*
- *Finger To Nose*
- *Open/Close Hand*
- *Tremor*
- *Pronation/Supination*

Each participant was asked to perform the tasks in the order above at 3 different speeds, each repeated 2 times.
For the *Finger To Nose* and *Tremor* tasks, speed was not considered relevant for the experiment.
The speeds were marked by means of a digital metronome set to:

- Slow (S): 75 bpm
- Normal (N): 115 bpm
- Fast (F): 140 bpm

The use of the metronome was integrated as a reference for the subject, helping him to keep a constant rhythm in the execution of the task.
The duration of each task was set to 20s, at the end of which the subject could rest his arm while the operators verified that all the recordings had been generated and saved correctly. Any movements of the markers were performed in this time interval.
Once the 22 recordings for the right hand were completed, the entire procedure was repeated for the left hand, ensuring that the markers were still attached to the

skin after the move.
If one of these had come off during the execution of a test or one of the cameras had not started, the task was repeated.
At the end of the experiment, it was checked that all 44 recordings had been generated and saved correctly and the brightness of the room was measured using a luxmeter, information that was correlated with the dimensions of the subject's hand.
The entire procedure lasted on average 40 minutes per person.

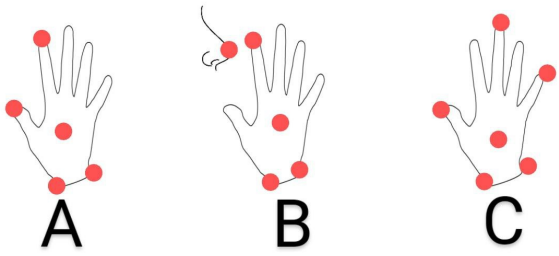


Figure 4.6: Marker Configuration: A for Finger Tapping, B for Finger to Nose, C for Open/Close, Tremor and Pronation/Supination.

4.2.1 Tasks

The tasks analyzed in this experiment are commonly used for the assessment of Parkinson's disease in the MDS-UPDRS scale (Movement Disorder Society - Unified Parkinson's Disease Rating Scale) [14].
The aim of these gestures is to evaluate the motor function, speed and coordination of a patient, allowing to detect and monitor over time the progression of 3 Parkinson's symptoms: *bradykinesia*, *rigidity* and *tremor* [15].
They will be described below:

- **Finger Tapping (FT):** consists in the simultaneous movement of thumb and index finger, from an initial open position (thumb and index finger distant) to a final closed position (thumb and index finger touching) alternatively. This task allows to measure the slowness of the movement as well as the amplitude and loss of rhythm due to a possible deterioration of motor coordination.

- **Finger To Nose (NOSE):** This test consists of touching your nose and a target placed in front of the patient (often the examiner's finger). In our case, the target was not present but the patient was asked to: fully extend the arm towards the camera placed in front (the DVXplorer) and bring the index finger as close as possible to the nose without touching the marker positioned there.

- **Open/Close Hand (OC):** consists of opening and closing the hand, until it forms a fist. In this task, the subjects were instructed not to force the closure too much so as not to compromise the visibility and adherence of the positioned markers.
- **Tremor (TR):** consists of fully extending the arm at shoulder height in front of the camera. To make the assessment more realistic, the subjects were asked to keep their eyes closed for the entire duration of the task so as to make any movement involuntary.
- **Pronation and Supination (PS):** consists of the alternating movement of the hands from a pronation position (palm facing the camera) to a supination position (palm facing the subject). This task highlights not only bradykinesia, but also potential signs of rigidity and difficulty in motor coordination.

4.3 Data Collection Process

Once the protocol was explained, the markers were positioned and the cameras were confirmed to be ready and capturing the entire scene, we moved on to the actual data acquisition process.
The bash scripts for the two Kinects were launched from two different terminals, invoked in this way:

```
» REGISTRATORE_KINECT_1   X01_R   FTS1
» REGISTRATORE_KINECT_2   X01_R   FTS1
```

Where the individual fields have the following meaning:

- `RECORDER_KINECT_X`: indicates the script used by kinect 1 and 2 to record data. The script automatically creates a folder, named based on the subject and the current task, in which it will save the completed recording. Once launched, the script will put the relative kinect in waiting state for the master.
- `X01_R`: represents the subject's identifier and the hand used R (right), L (left).
- `FTS1`: specifies the type of task recorded (FT, NOSE, OC, TR, PS), the speed (S, N, F) and the test (1 or 2).

Similar procedure for the DVXplorer: at the end of each task, if necessary, the current folder was manually changed and the next recording was named.
The notation followed was:

```
E:\DVX_Recs\X03\X03_LEFT\OC\dvSave_OCF1.aedat4
```

With the kinects waiting, the metronome was started and the subject began to perform the current task.
At this stage, operator A initiated the OptiTrack recordings (along with the Kinect devices), while operator B started the DVXplorer recordings.
The synchronization of the start was achieved through a loud verbal countdown. After 20s, the recordings were automatically ended and saved in the appropriate folders with the correct names. A summary of all the task combined with the different speed is reported in Tab.4.1.

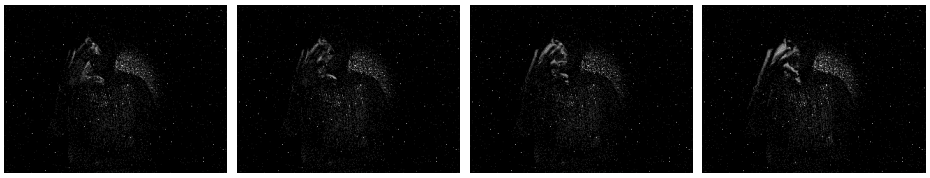
Task	**S (Slow)**	**N (Normal)**	**F (Fast)**
FT	FTS1, FTS2	FTN1, FTN2	FTF1, FTF2
NOSE	/	NOSE1, NOSE2	/
OC	OCS1, OCS2	OCN1, OCN2	OCF1, OCF2
TR	/	TR1, TR2	/
PS	PSS1, PSS2	PSN1, PSN2	PSF1, PSF2

Table 4.1: Task combinations with different speeds (S, N, F), repeated twice for each combination (22 tasks per hand).

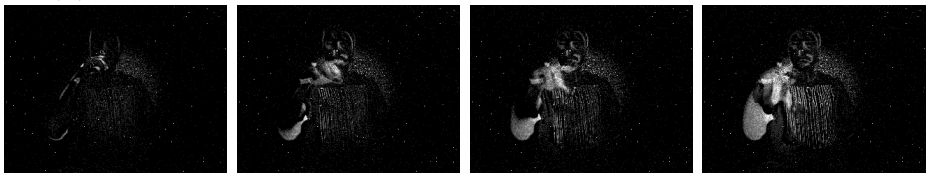
4.4 Dataset Organization

The resulting dataset can be summarized with the following numbers:

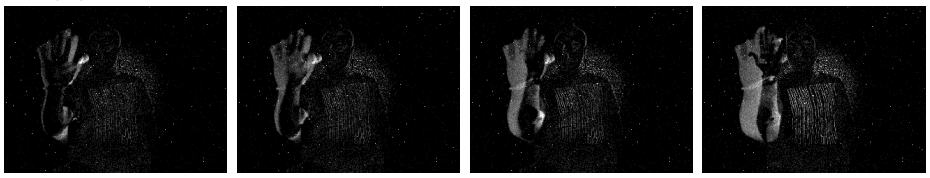
- **Participants**: **25 subjects** performed **5 tasks**, each at 3 different speeds and repeated 2 times for each combination.
- **Recordings**: for each participant, 22 recordings were made per hand, for a total of 44 recordings per subject and **1100 total recordings** per camera.
- **Duration**: each task lasted **20 seconds**, accumulating a total of over **6 hours of recordings**.
- **DVXplorer dataset size**: **73 GB**, containing all the data acquired by the DVXplorer.
- **Total size**: **6 TB**, considering the entire set of recordings from all cameras (DVXplorer, Kinect and OptiTrack).



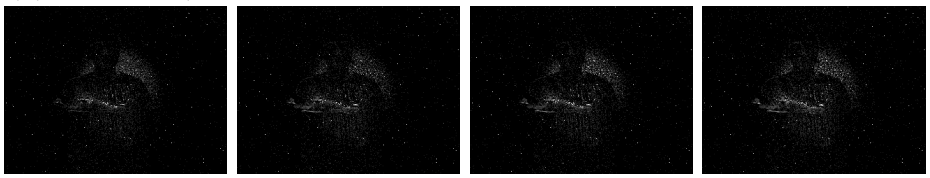
(a) Finger Tapping: sequence of 4 frames accumulated at 30 fps.



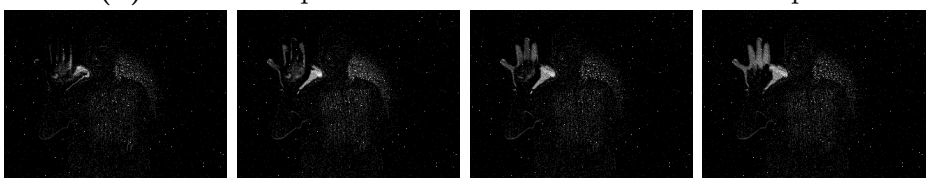
(b) Finger To Nose: sequence of 4 frames accumulated at 30 fps.



(c) Pronation/Supination: sequence of 4 frames accumulated at 30 fps.



(d) Tremor: sequence of 4 frames accumulated at 30 fps.



(e) Open/Close Hand: sequence of 4 frames accumulated at 30 fps.

Figure 4.7: Sequence of gestures represented by 4 frames each. All sequences are accumulated at 30 fps.

Chapter 5

Training SNN for Gesture Recognition

Training an artificial neural network to recognize gestures, with good accuracy, is a complex process as it involves multiple steps, iterated over and over again.
First, it is essential to ensure that the data provided as input to the network are compatible and interpretable by the chosen model. This involves applying transformations and preprocessing to the raw data collected.
Next, the network model must be defined, specifying its layers, neuron types, learning methods, parameters, and the metrics to monitor the learning process.
Then comes the training and validation process, in which the chosen model is trained using the preprocessed data and validated on new data, in order to measure its learning and ability to generalize on data never encountered before.
In this chapter, the various steps will be examined in order, from the pre-processing of the dataset, introducing the *Tonic* library, to the definition of the network using *snnTorch*. Finally, it will be illustrated how the hyperparameter search process was carried out and optimized using *wandb*.

From this section onwards, the analysis will be focused exclusively on the neuromorphic version of the dataset. Data acquired through Kinect and OptiTrack will no longer be considered, as the goal is to train an SNN using exclusively neuromorphic data.

5.1 Data Preprocessing

Data preprocessing is the set of techniques and operations applied to the raw data of a dataset to best prepare them for training a ML model.

In the case of event-based datasets, this process is particularly important to eliminate spurious events caused by sensor noise, reduce the spatial/temporal dimension and extract regions of interest, reducing computational costs and any superfluous data.
Furthermore, it is common practice to accumulate events in more classic formats, such as frames or voxel grids, to be able to exploit the consolidated architectures and techniques of convolutional neural networks.
For this purpose, **Tonic** comes to the rescue, a python library specifically developed to work with neuromorphic datasets. Among the main features offered are data preprocessing and representation techniques, ready-to-use neuromorphic datasets, provided with related classes, and dataloaders optimized to work with large neuromorphic datasets.

5.1.1 From aedat to numpy

A first change made to the entire dataset was the conversion of the recording formats from *aedat4* to *numpy*.
Tonic does not recommend working directly with aedat for reasons of efficiency and practicality due to complex formatting, high computational weight and above all for reasons of compatibility with machine learning models and third-party libraries. Numpy files solve these problems by offering faster loading, greater compatibility and better storage.
A python script was therefore developed that can automatically explore each recording in the starting dataset and convert it to the desired format.
The script keeps the original structure of the dataset intact, including folders and subfolders, also preserving the file names to ensure consistency and ease of access.

Event example in *aedat4*:

$$\text{Timestamp: } 1234567890, \quad \text{X: } 35, \quad \text{Y: } 40, \quad \text{Polarity: } 1$$

Array *NumPy* of events:

$$\begin{bmatrix} \text{Timestamp} & \text{X} & \text{Y} & \text{Polarity} \\ 1234567890 & 35 & 40 & 1 \\ 1234567891 & 36 & 41 & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

5.1.2 Recordings Split

The dataset was acquired by recording each task for a duration of 20 seconds, with each recording representing the subject performing a specific task for the entire

period.
To increase the amount of data available for training the model, it was decided to split each 20-second recording into 10 shorter recordings, each lasting 2 seconds.
A python script was then developed that transforms the original 1100 recordings of the dataset, already converted to NumPy format, into 11000 recordings of 2 seconds each.
The script keeps the original structure of the dataset intact, including folders and subfolders while the resulting recordings were renamed as follows:

- Original recording name: `dvSave_FTS1.npy`
- Generated segment names:
  - `dvSave_FTS1_segment_1.npy`
  - `dvSave_FTS1_segment_2.npy`
  - ...
  - `dvSave_FTS1_segment_10.npy`

In this way the task label is kept intact, which is fundamental in the extraction phase for the calculation of accuracy and loss metrics.
After the splitting is completed, the resulting dataset can be summarized as follows:

- **440** recordings per subject.
- **25** subjects.
- **11000** total recordings.

5.1.3 Dataset Split

The resulting dataset, after being converted and divided as described in the previous paragraphs, was then split into *Training Set* and *Test Set*, respectively used to train and test the model.
It is essential to underline that the **Test Set** was used exclusively at the end of the training process, as a final phase to evaluate the model's ability to generalize on data never seen during training. In this way it was possible to obtain a realistic evaluation of the model's performance on new data and not related to those already used during training.
Furthermore, the division of subjects between the *Training Set* and the *Test Set* was made to be as uniform as possible, in terms of the size of the subjects' hands and the lighting conditions present during data collection.
To this end, the data collected during the creation of the dataset were used, regarding the size of the subject's hand and the lighting measured in the laboratory

		Left			Right			
Subject	Lumen	p [cm]	h [cm]	Palm Area	p [cm]	h [cm]	Palm Area	Media Area
1	220	11	17.5	192.5	11	17.5	192.5	192.5
2	152	12.5	19.5	243.75	12.5	19.5	243.75	243.75
3	195	12	20.5	246	12	20.5	246	246
4	200	12	19.5	234	12	19.5	234	234
5	175	11.5	18.5	212.75	11.5	18.5	212.75	212.75
6	185	12	18.5	222	12.5	18	225	223.5
8	218	11.5	19	218.5	11.5	19	218.5	218.5
9	172	11.5	18.5	212.75	11.5	18.5	212.75	212.75
10	179	11.5	17.5	201.25	10.5	17.5	183.75	192.5
11	230	11.5	18.5	212.75	12	18.5	222	217.375
12	145	13	20.5	266.5	13.5	20.5	276.75	271.625
13	175	11	17	187	11	17	187	187
14	150	11.5	18.5	212.75	11.5	18.5	212.75	212.75
15	195	12	19.5	234	12.5	19.5	243.75	238.875
16	186	11.5	18.5	212.75	11.5	18.5	212.75	212.75
17	184	13	21	273	13.5	21.5	290.25	281.625
18	185	12	18.5	222	12	18.5	222	222
19	205	12	19	228	12.5	19	237.5	232.75
20	185	11	17.5	192.5	11	17.5	192.5	192.5
21	203	12.5	20.5	256.25	12.5	20.5	256.25	256.25
22	208	12.5	19.5	243.75	12.5	19.5	243.75	243.75
23	184	12.5	19.5	243.75	12.5	19.5	243.75	243.75
24	180	11.5	18.5	212.75	11.5	18	207	209.875
25	184	12.5	19.5	243.75	13	19.5	253.5	248.625
26	193	12.5	19.5	243.75	12.5	18.5	231.25	237.5

Figure 5.1: Lab luminosity and hand size per subject

at the time of the acquisitions Fig.5.1.

The size of each subject's hand was obtained by recording two main dimensions: the distance from the tip of the middle finger to the beginning of the wrist, indicated by h, and the distance between the beginning of the little finger and the beginning of the thumb, indicated by p, as illustrated in Fig.5.2. The surface of the palm was

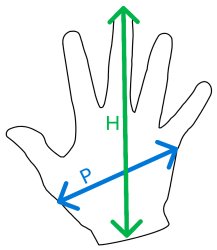


Figure 5.2: Hand Sizes

calculated approximately as:

$$\text{PalmArea} = p \cdot h \quad [\text{cm}^2]$$

and was averaged for the right and left hand to obtain *Median Area*.
With this information, a scatter plot with subject marker was then created, relating the brightness and surface values of the hand for each subject Fig.5.3.

Analyzing the graph, the dataset was divided according to the proportion 70% Training and 30% Test. The division was carried out in order to obtain a balanced

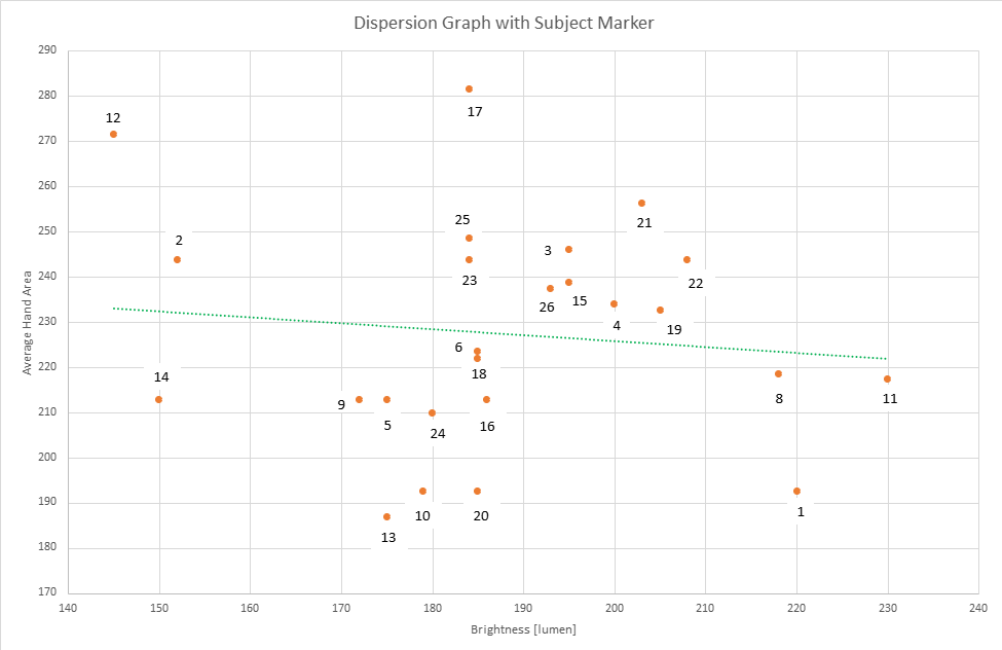


Figure 5.3: Dispersion Graph with subject markers

representation of the subjects in all areas of the graph. In Tab.5.1 can be observed the division of the subjects.

Training Set	**Test Set**
1, 3, 5, 6, 7, 8, 9, 10, 12, 13, 14, 16, 19, 21, 22, 23, 24, 25	2, 4, 11, 15, 17, 18, 20

Table 5.1: Distribution of subjects between Training Set and Test Set.

5.1.4 Tonic Transformations

To make the input events to the model more understandable, a series of transformations, belonging to the Tonic library, have been applied.
The transformations and their parameters have been defined starting from the size of the neuromorphic sensor *320*x*240* and will be briefly described.

Denoise: this transformation removes noisy events that occur in isolation in time and space. The *filter_time* parameter specifies a time window within which the events are considered valid, in our case set to 10000 µs (10 ms).

CenterCrop: reduces the spatial dimensions by focusing on the most significant area, the center in our case. A central crop is then performed on the event matrix in order to obtain a final resolution of *200*x*200*.

Downsample: performs a further resizing of the cropped area bringing it to a lower resolution of *64*x*64*.

ToFrame: this transformation accumulates the sequence of events along time, so as to obtain 60 frames resulting from a 2 s recording. The representation in frames was chosen because it adapts very well to convolutional layers, useful for feature extraction during the model training process.

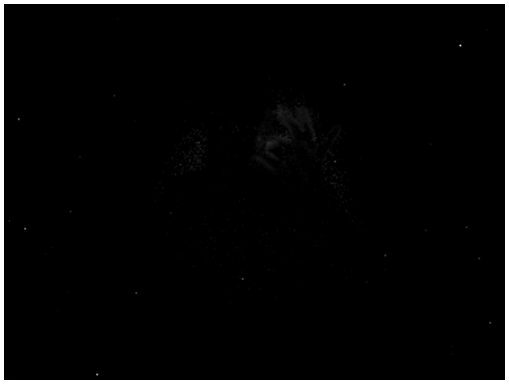
In Fig.5.5 the transformation process after each single transformation.

5.1.5 Applying Transformation and Caching

Tonic allows to work with some of the most popular neuromorphic datasets, such as N-MNIST and DVSGesture, each with a specific *Dataset* class. These classes provide an interface to load, manage and transform the related event datasets.

Since our dataset was created from scratch, it was necessary to define a custom class called *DVXdataset*, inspired by the DVSGesture dataset class, which is the most similar to ours. This new class includes a custom function dedicated to extracting labels directly from the name of the recordings, respecting their notation. Furthermore, being compatible with the PyTorch DataLoader, the class allows to perform batching (splitting the data into batches), shuffling (mixing the data) and applying transformations, all simultaneously with the training phase.

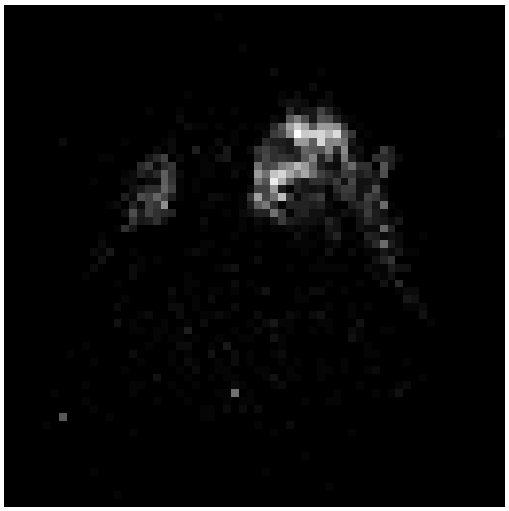
To simplify the application of transformations to the whole dataset, a separate script has been developed. This script has the exclusive task of loading the training and test data, applying the defined transformations and saving each transformed



(a) No transformation

(b) After Denoising

(c) After CropCenter



(d) After Downsample

Figure 5.4: Transformations sequence: original data, denoised, cropped, and downsampled (1 frame illustrated).

record in cache in the hdf5 format, using the DataLoader.
The conversion to *hdf5* format is commonly used for managing large amounts of data as it allows rapid access, is optimized for parallelism and allows persistent saving, avoiding repeating the transformation process. At the end of the process the entire transformed dataset was correctly saved on the lab server, ready to be used.

Listing 5.1: Cached Dataset

```
DVXplorer_Dataset/
                    train/
                        0.hdf5
                        ...
                        7920.hdf5

                    test/
                        0.hdf5
                        ...
                        3080.hdf5
```

5.2 Model Architecture

Once the pre-processing phase was completed, we moved on to the definition of the Spiking Neural Network (SNN).
The models described below are based on the Pyhton library called *snnTorch*, designed to simplify the construction and training of SNNs using the PyTorch framework. This library allows the use of neuron models plausibly similar to their biological counterparts, such as the *Leaky Integrate and Fire* abbreviated LIF.
The LIF neuron has the ability to model temporal dynamics thanks to its intrinsic temporal memory (membrane potential and decay). This allows the network to temporally correlate consecutive frames, which contain partial information of the gesture, and to recognize patterns that emerge in the sequence.
This neuron model has been used in all the networks presented below and is described in chapter 3.
In addition to the description of the tested networks, this chapter will illustrate the training, validation and search process for the best hyperparameters.

5.2.1 Original Net

As a starting point, the network described in Tab.5.2 was tested, whose structure is inspired by a model already validated and correctly trained on the DVSGesture dataset.
The only changes made concern the fully connected layer, increasing the number of

neurons from 800 to 5408 and reducing the output neurons from 11 to 5, like the classes of our dataset.
The network receives 64x64 frames as input, extracts spatial features through the two convolutional layers and finally maps the extracted features into final predictions thanks to the fully connected layer.
The LIF neuron also allows capturing temporal dependencies between the various frames. The activation threshold *th*, i.e. the value of the membrane potential beyond which the neuron activates, was set to 1 (standard value). While β, the decay rate of the membrane potential over time, has been set to 0.5 which leads to a 50% decay in the potential value between one step and the next.

Layer	**Type**	**In Dim**	**Out Dim**	**Params**
1	Conv2D	$[2, 64, 64]$	$[12, 60, 60]$	Kernel 5×5
2	MaxPool	$[12, 60, 60]$	$[12, 30, 30]$	Pool 2×2
3	LIF	$[12, 30, 30]$	$[12, 30, 30]$	$\beta = 0.5$, $th = 1.0$
4	Conv2D	$[12, 30, 30]$	$[32, 26, 26]$	Kernel 5×5
5	MaxPool	$[32, 26, 26]$	$[32, 13, 13]$	Pool 2×2
6	LIF	$[32, 13, 13]$	$[32, 13, 13]$	$\beta = 0.5$, $th = 1.0$
7	FullyConn	$[5408]$	$[5]$	Output=True
8	LIF	$[5]$	$[5]$	$\beta = 0.5$, $th = 1.0$

Table 5.2: Original Net Structure.

5.2.2 Simplified Net

The second model, implemented and tested, is grafted onto the previously explained network, simplifying its complexity especially related to the last fully connected layer.
Having a high number of neurons in input to the last layer, as we will see, can lead to *Overfitting* problems, that is, the model adapts too well to the data with which it is trained but is not able to generalize on new ones.
The *Simplified Net* reduces the complexity by using pre-trained components for the initial convolutions and introducing a new convolutional layer and 2 fully connected ones.
The initial convolutional layers, trained on the DVSGesture, are frozen, that is, the weights are not updated during the training process. The structure of the network is summarized in Tab.5.3

Layer	Type	In Dim	Out Dim	Params
1	Conv2D	$[2, 64, 64]$	$[12, 60, 60]$	Kernel 5×5
2	MaxPool	$[12, 60, 60]$	$[12, 30, 30]$	Pool 2×2
3	LIF	$[12, 30, 30]$	$[12, 30, 30]$	$\beta = 0.5$, $th = 1.0$
4	Conv2D	$[12, 30, 30]$	$[32, 26, 26]$	Kernel 5×5
5	MaxPool	$[32, 26, 26]$	$[32, 13, 13]$	Pool 2×2
6	LIF	$[32, 13, 13]$	$[32, 13, 13]$	$\beta = 0.5$, $th = 1.0$
7	Conv2D	$[32, 13, 13]$	$[64, 12, 12]$	Kernel 3×3
8	MaxPool	$[64, 12, 12]$	$[64, 6, 6]$	Pool 2×2
9	LIF	$[64, 6, 6]$	$[64, 6, 6]$	$\beta = 0.5$, $th = 1.0$
10	FullyConn	$[2304]$	$[128]$	Weights $2304 \rightarrow 128$
11	LIF	$[128]$	$[128]$	$\beta = 0.5$, $th = 1.0$
12	FullyConn	$[128]$	$[5]$	Weights $128 \rightarrow 5$
13	LIF	$[5]$	$[5]$	$\beta = 0.5$, $th = 1.0$, Output=True

Table 5.3: Simplified Net Structure.

5.2.3 Dynamic Net

The *Simplified Net* turned out to be a simpler network than the original, but it had an opposite problem known as *Underfitting*: the network was unable to learn meaningfully from the input data.
Although this model did not produce the expected results, together with the original *Original Net* model, it allowed to identify the two extremes of complexity in the network design.
A new dynamic model *Dynamic Net* was then developed, capable of exploring the search space between the two extremes. The goal was to identify the optimal number of neurons, balancing the complexity to avoid both overfitting and underfitting. To this end, a network was defined whose number of neurons, at the input of the last fully connected layer, was dynamic. Furthermore, a dropout layer was added between the later fully connected layers, whose task is to deactivate a percentage of the neurons, taken randomly, at each forward pass. In this way, the learning process is more robust and less subject to overfitting.
The network is summarized in Tab.5.4. As can be seen from the table, some dimensions are expressed as variables: *neurons_fc1* and *dropout_rate*.
The values assumed by these were set as search parameters in the model optimization process, using *wandb*. The search will be explained in the next paragraphs but it is anticipated that the ranges were:

$$neurons\_fc1 \in [128, 500], \quad dropout\_rate \in [0.2, 0.3, 0.4, 0.5]$$

Layer	Type	In Dim	Out Dim	Params
1	Conv2D	$[2, 64, 64]$	$[12, 60, 60]$	Kernel 5×5
2	MaxPool	$[12, 60, 60]$	$[12, 30, 30]$	Pool 2×2
3	LIF	$[12, 30, 30]$	$[12, 30, 30]$	$\beta = 0.5$, $th = 1.0$
4	Conv2D	$[12, 30, 30]$	$[32, 26, 26]$	Kernel 5×5
5	MaxPool	$[32, 26, 26]$	$[32, 13, 13]$	Pool 2×2
6	LIF	$[32, 13, 13]$	$[32, 13, 13]$	$\beta = 0.5$, $th = 1.0$
7	Conv2D	$[32, 13, 13]$	$[64, 11, 11]$	Kernel 3×3
8	MaxPool	$[64, 11, 11]$	$[64, 5, 5]$	Pool 2×2
9	LIF	$[64, 5, 5]$	$[64, 5, 5]$	$\beta = 0.5$, $th = 1.0$
10	FullyConn	[1600]	[neurons_fc1]	Weights $1600 \rightarrow$ neurons_fc1
11	LIF	[neurons_fc1]	[neurons_fc1]	$\beta = 0.5$, $th = 1.0$
12	Dropout	[neurons_fc1]	[neurons_fc1]	$p =$ dropout_rate
13	FullyConn	[neurons_fc1]	[5]	Weights neurons_fc1 $\rightarrow 5$
14	LIF	[5]	[5]	$\beta = 0.5$, $th = 1.0$, Output=True

Table 5.4: Dynamic Net Structure.

5.3 Training

In this paragraph, the training process, common to all network models trained in this project, will be described.
The goal was, starting from the train dataset, to train a subset of users *train_subset* and validate the model on another subset *val_subset*, to evaluate its generalization ability during learning.
This process is called *Validation* and will be illustrated in more detail in the next paragraph.
Each generated subset was loaded via a special *DataLoader*, having set the number of samples per batch *batch_size* to 32 and enabled *shuffle* in order to mix the data at each epoch.
The number of epochs *n_epochs*, i.e how many times the entire train_subset is passed through the model during training, was varied based on the simulations between 100 and 200.
To simplify the feature extraction process, the weights of a model already trained on the DVSGesture dataset were loaded and transferred to the first two convolutional layers of the instantiated network, which were then frozen.
Adam was chosen as the optimizer, configured with variable *learning_rate* and *weight_decay*, as they belong to the search space too. The ranges of values that can be assumed by these parameters were included in *wandb*.
To refine the learning as the epochs passed, a *scheduler* was instantiated, responsible for reducing the *lr* by a factor γ every so many epochs.

To calculate the loss, the *Cross Entropy Loss* was used as it is well suited to the nature of the problem. The loss is calculated on all the time steps and samples of the batch, quantifying the discrepancy between the membrane potentials, returned after the forward, and the correct labels.
During the training loop the *forward* function, belonging to the instantiated net class, receives as input a batch of 32 recordings. Each recording is composed of temporal sequences of 60 frames, representing snapshots of the same gesture, which are processed in an orderly fashion by the network. At the end of the sequence the function returns the membrane potentials for each class accumulated over time that will then be used to calculate the loss and accuracy.
At this point the gradients related to the previous batch are deleted and the new gradients of the loss are calculated with respect to the weights of the network, using the *backpropagation* technique. Finally the weights of the network are updated.

5.4 Validation of the Model

The validation process consists of evaluating the model on a separate dataset, called *val_subset*, that was not used during training. This step aims to monitor metrics such as accuracy and loss (*loss*) during training, providing an estimate of how well the model generalizes to unseen data.
It is important to distinguish validation from the testing process: the *Test dataset* is in fact used exclusively at the end of training and does not affect training.

In our case, the 18 subjects of the *Training dataset* were split into two subsets: 12 subjects for the *train_subset* and 6 for the *val_subset*. During each epoch, the model was trained on the *train_subset* and validated on the *val_subset*, recording the respective *performance metrics* to monitor the training progress.
However, this approach is sensitive to the specific data split. If the *val_subset* does not adequately represent the distribution of the complete dataset, the obtained metrics may be inaccurate.
To overcome this problem, the *Cross Validation* technique was implemented, which will be explored in the following paragraph.

5.4.1 Cross Validation

Cross Validation is a technique that splits the dataset into K *folds*, used in turn for training and validation of the model.
In each iteration, the model is trained on $K-1$ folds and validated on the remaining fold, rotating between the folds used for training and validation.
During the process, validation metrics are recorded for each fold and, at the end of all iterations, they are averaged to provide an overall estimate of the robustness of

the model.
In our case, we decided to implement a *3-Fold Cross-Validation* (3CV), that is, the 18 subjects were split into 3 folds, each containing 6 subjects. An example of subdivision is the following:

$$\text{Fold}_1 = \{S_1, S_2, S_3, S_4, S_5, S_6\}$$

$$\text{Fold}_2 = \{S_7, S_8, S_9, S_{10}, S_{11}, S_{12}\}$$

$$\text{Fold}_3 = \{S_{13}, S_{14}, S_{15}, S_{16}, S_{17}, S_{18}\}$$

$$\begin{aligned}
&\text{Iteration 1:} \quad \text{train_subset} = \text{Fold}_2 \cup \text{Fold}_3, \quad \text{val_subset} = \text{Fold}_1 \\
&\text{Iteration 2:} \quad \text{train_subset} = \text{Fold}_1 \cup \text{Fold}_3, \quad \text{val_subset} = \text{Fold}_2 \\
&\text{Iteration 3:} \quad \text{train_subset} = \text{Fold}_1 \cup \text{Fold}_2, \quad \text{val_subset} = \text{Fold}_3
\end{aligned}$$

5.5 Hyperpameters Optimization

Training a network is a very complicated process given the large number of parameters to configure.
Different combinations of parameters significantly affect both the speed of convergence and the ability of the model to generalize.
Finding the right combination of these to maximize performance therefore requires numerous iterations.
To facilitate this process, *Weights&Biases* has been integrated into the code, a platform that allows to automate the search for hyperparameters through a tool called *Sweep*.
A Sweep is an iterative process that explores different combinations of hyperparameters, called *runs*, applies them to the model and analyzes the results, in order to maximize/minimize a specific metric of our interest.
The search ranges of these parameters must be set within a python dictionary named *sweep_config* containing:

- The method of exploration of the search space
- The goal to achieve
- The parameters to explore to reach the objective

The search method indicates how the hyperparameters of each run should be chosen, either randomly (random search), exhaustively (grid search), or in a more optimized way (bayesian search). In our case, we decided to opt for a bayesian search, an advanced search method, which builds a probabilistic model to try to find the optimal combination, based on the objective metric of the previous runs.

The choice of this method derives from the numerous hyperparameters and the consequent large search space; an approach of this type allows us to direct the search on regions of our interest without exploring all the possible combinations. The objective instead indicates the metric that we want to optimize, minimizing or maximizing it. It is very important because it guides the exploration process of the combination space, especially in the case of bayesian search.
For our experiments, the validation loss *val_loss* was set trying to minimize it. Optimizing on this metric is a common practice because it gives a good idea of how the model is able to generalize on new data. However, we also tried to maximize the validation accuracy and the balanced accuracy, the latter being particularly useful in the case of unbalanced classes (the NOSE and TR classes contain fewer samples than the others).

```
sweep_config = {
    "method": "bayes",                      # Optimization Method
    "metric": {                             # Metric to optimize
                "goal": "minimize",
                "name": "val_acc"
    },
    "parameters": {                         # Parameters to explore
                    "lr": {
                        "min"   : 0.0001,
                        "max"   : 0.001
                    },
                    "neurons_fc1": {
                        "min"   : 128,
                        "max"   : 500,
                        "distribution": "int_uniform"
                    },
                    "weightdecay": {
                        "values": [1e-6, 1e-5, 1e-4, 1e-3]
                    },
                    "dropout": {
                        "values": [0.2, 0.3, 0.4, 0.5]
                    }
                    "lr_decay_epcs": {
                        "values": [5, 10, 20]
                    }
    }
}
```

The search parameters were as follows:

- **Learning rate:** represents the rate at which the model updates its weights during the learning process, influencing how quickly the model learns.
- **Neurons_fc1:** indicates the number of neurons in the first fully connected layer, increasing or reducing the complexity of the network.
- **Weight decay:** is used to regularize the weights of the network, penalizing weights that are too large.
- **Dropout:** the probability that neurons are dropped (turned off) during training was indicated.
- **Lr_decay_epcs:** indicates after how many epochs to decay the learning rate.

The number of neurons in the fully connected layer and the dropout are parameters that directly influence the structure and behavior of the network, for this reason they are called *intrinsic parameters*.
The number of runs, and consequently the number of different combinations applied to the model, is configurable a priori and has been varied based on the experiment in progress (usually between 100 and 200).
A summary diagram of the described pipeline can be observed in Fig.5.5.

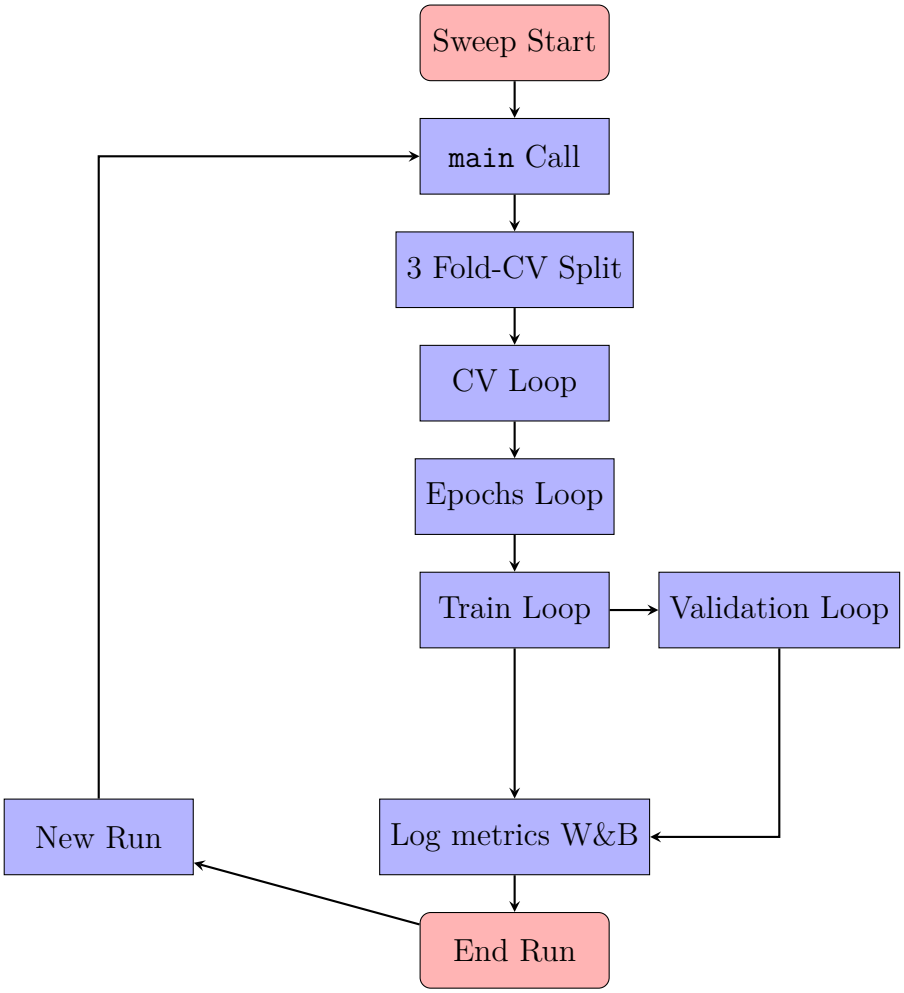


Figure 5.5: Hyperparameter optimization pipeline

Chapter 6

Experimental Results

In this section, the results obtained following the simulations of the previously described models will be presented.
The structure used for training, validation and optimization of the parameters was common to all the models and can be summarized with the pipeline presented in Fig.5.5.
The intrinsic parameters of the networks and the ranges of the hyperparameters have been modified based on the results learned from the previous simulations.

6.1 Original Net Performance

Original Net was the starting model.
This network has been successfully tested on DVSGesture, in previous projects to this one, and the weights file was available for use.
The changes made concern the last fully connected layer, increasing the input neurons from 800 to 5408 and reducing the output neurons to 5, like the classes of our dataset.
The resulting structure is observable in Tab.5.2.

After loading the weights of the trained model, the network was instantiated by freezing the first 2 convolutional layers so that the weights were not updated during training.
The only trainable layer was therefore the fully connected one.
The results of the model are observable in Fig.6.1.
Since the first epochs the model has performed anomalously, showing an extremely high train accuracy that reached 100% after only 4/5 epochs. The validation accuracy remained low and the relative loss increased as the epochs passed.

This behavior was repeated for most of the runs generated by sweeps, regardless of the combinations of parameters tested and allowed us to understand that the model was in *overfitting* mode, that is, it was not learning from the data but rather memorizing them.
This usually occurs when the defined network is too complex compared to the task to be performed or the dataset used.
We therefore came to the conclusion that 5406 neurons were too many and we proceeded to simplify the network.

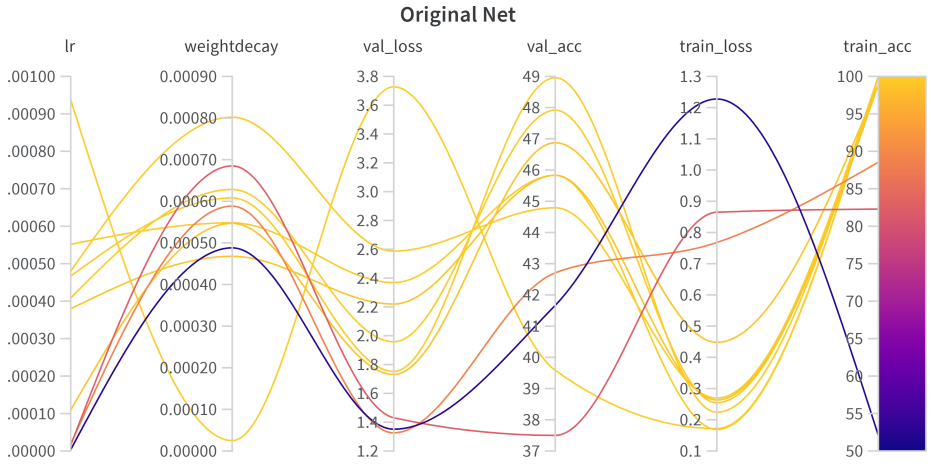


Figure 6.1: Original Net Performance

6.2 Simplified Net Performance

Following the analysis of the performance of the Original Net, we moved on to define a less complex network *Simplified Net*. This new model is grafted onto the previous one, keeping the two convolutional layers unchanged, provided with the starting weights, which remain frozen throughout the training process.
The changes made concern the introduction of a third convolutional layer (trainable) and two fully connected layers (trainable).
To avoid overfitting problems, 128 input neurons were assigned to the last fc layer, mapped on the 5 output neurons representing the classes of the dataset.
The complete structure of the network is represented in figure Tab.5.3.
Looking at the simulation logs, it was immediately possible to notice how the model metrics were static despite the passing of the epochs.
The train accuracy as well as the validation accuracy remained at 27%, regardless

of the values assumed by learning rate and weight decay, for more than 100 epochs Fig.6.2.
It was therefore concluded that the network was in full *underfitting* mode, a situation completely opposite to that which occurred in the first model.
The instantiated network, lacking sufficient complexity, is unable to detect meaningful patterns necessary for making accurate predictions.
This experiment, however, allowed us to understand and define the two extremes

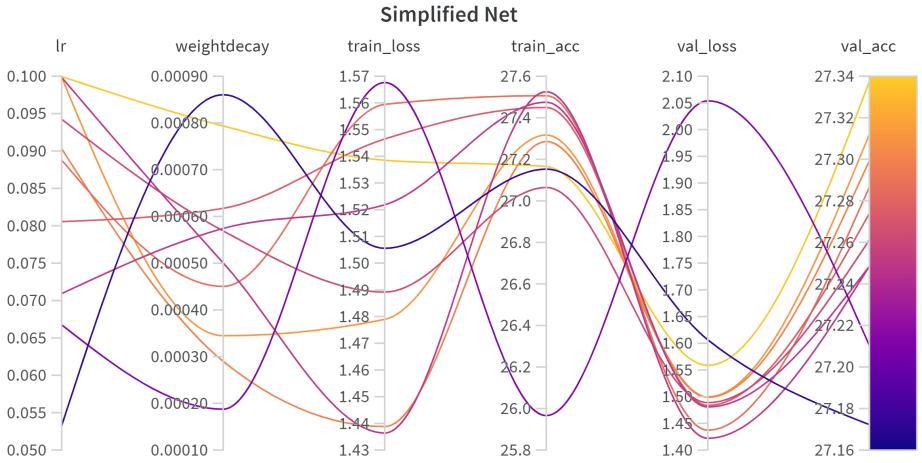


Figure 6.2: Simplified Net Performance

of complexity of the network, near which the model is in overfitting and underfitting regime. The search space regarding the number of neurons needed in the fully connected layers was therefore narrowed.
To confirm this analysis, a network with 500 neurons was defined. The results of the runs were encouraging, with train and validation accuracy increasing from epoch to epoch by a few percentage points but with a constant trend. The train accuracy reached the peak of 95%/97% only at the end of the experiment around epoch 100.
The validation accuracy stopped between 50% and 60% but with a very variable validation loss Fig.6.3.
These last results allowed us to further narrow the search field between 500 and 128 neurons, leading us to define the network *Dynamic Net*.

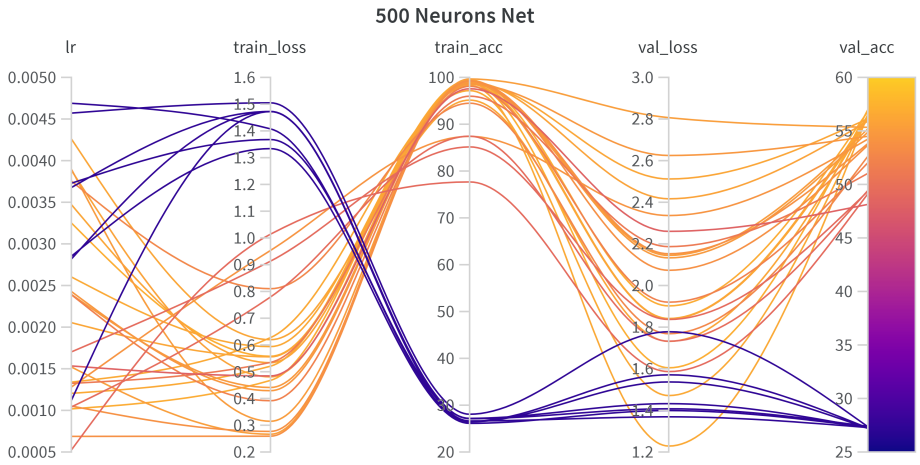


Figure 6.3: 500 Neurons Net

6.3 Dynamic Net Performance

The results of the performance analyses of previous models led to the definition of the *Dynamic Net*.
As the name itself suggests, it was decided to simulate the network on a *dynamic* number of neurons, variable from run to run, between 500 and 128.
Therefore, the following were added as search parameters to sweep:

- the number of neurons in layer fc1
- the dropout, specifying the values: [0.2, 0.3, 0.4, 0.5]
- the number of epochs, after which the learning rate is decreased: [5, 10, 20]

The graph in Fig.6.4, represents the results of 100 runs performed for 100 epochs. As can be observed, most of the runs tested by sweep converge on the same values of train accuracy and validation loss (gray curves). This is a bias of the sweep optimizer that, having to minimize the value of the validation loss, continues to try combinations of similar parameters that underfit.
However, if we focus on the colored curves, interesting combinations have been found. As can be observed, the train accuracy after 100 epochs is between 50% and 70% while the validation continues to decrease.
Considering the trend of the val loss over 100 epochs of some of these runs Fig.6.4, it seems that the network is learning little by little and still has room for improvement.

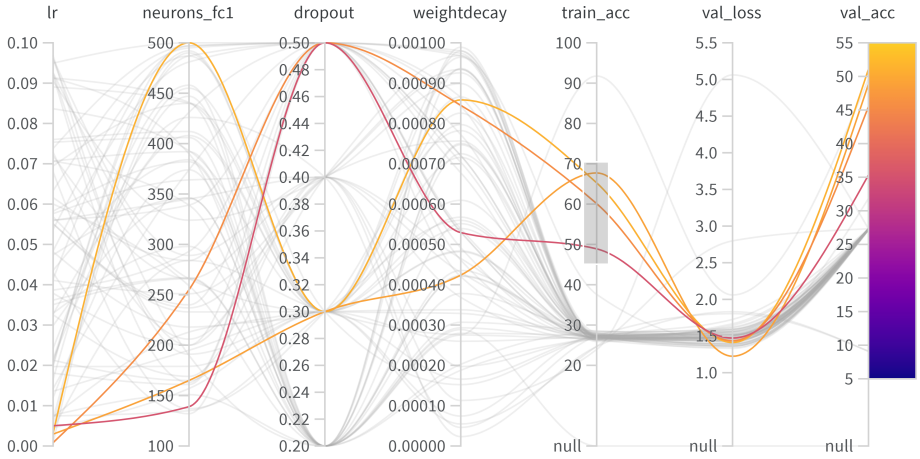
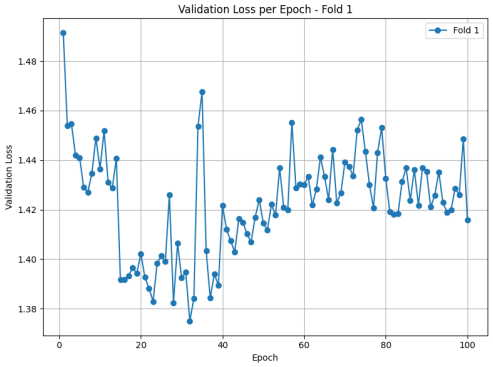
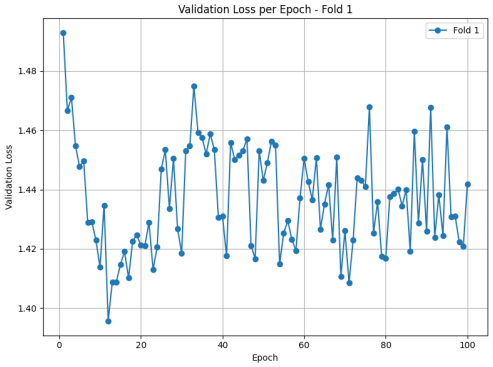


Figure 6.4: Dynamic Net Performance

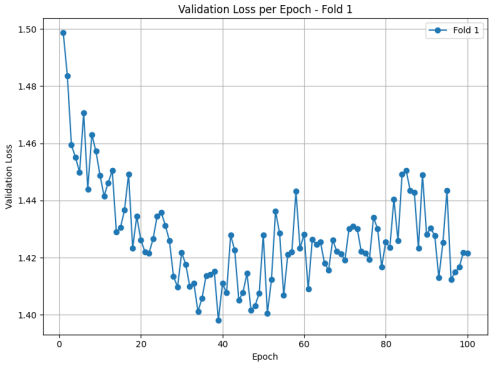
As a further attempt, the number of epochs was changed, increasing it to 200 and the second convolutional layer was unfrozen so that the relative weights were updated during training.
Furthermore, an early stopping mechanism was implemented for the runs that underfit so that the relatively low values of validation loss did not negatively influence the process of finding the best hyperparameters.
Validation accuracy values around 27% (+/- 3%) for more than 5 epochs indicate that the model, with the parameters of the current run, underfits. When this condition occurs, the run is stopped and the metrics are not logged on wandb, thus not contributing to giving a wrong direction in the exploration of the search space.
The results Fig.6.6 were promising: the underfitting runs did not contribute to the research while the validation accuracy reached values around 60% gradually increasing with the passing of the epochs. The validation loss is still at slightly high values but, observing the trend of this metric for some runs, it can be observed how the curve decreases over time Fig.6.7.



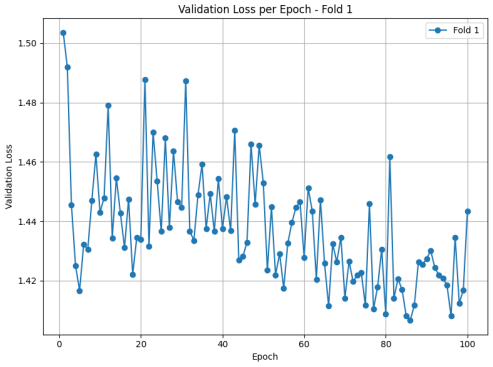
(a) Run 4



(b) Run 17



(c) Run 26



(d) Run 48

Figure 6.5: Validation loss trend over epochs

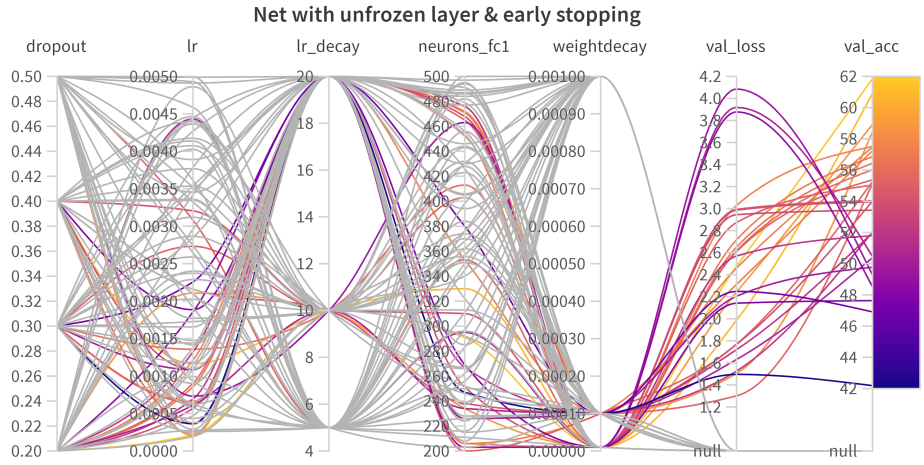


Figure 6.6: Unfrozen Net with early stopping on underfitting runs

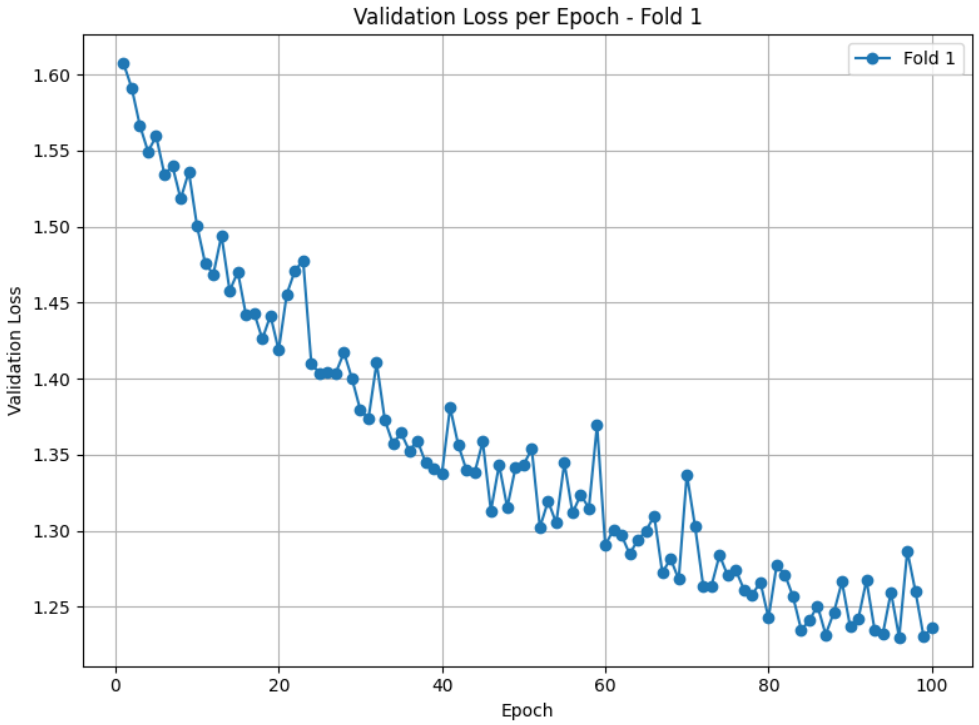


Figure 6.7: Promising run

Chapter 7

Discussion

7.1 Limitations of the Study

This project led to the creation of a large multimodal dataset of hand gestures, commonly used in the assessment and therapy of subjects affected by Parkinson's and neurodegenerative diseases.
However, it is important to underline that the participants in the experiment are healthy subjects and not actually affected by the disease.
Involving real patients would have been complicated for a number of reasons:

- **ethical and regulatory:** it would have been necessary to obtain authorizations from hospital institutions, a very long process.

- **medical supervision required:** a specialized medical figure would have had to supervise the entire data collection work, providing assistance to the patient if necessary, adding organizational complexity and costs.

- **logistical and organizational:** collecting a good number of patients would have required time and greater organization, also considering their transportation.

- **physical limitations:** these tasks, although designed to test the physical conditions of real patients, create fatigue especially if performed according to a protocol like ours.

However, this work, although not having clinical value, represents a fundamental preliminary step for future studies and research conducted with real patients.

7.2 Privacy and Energy Efficiency

The use of event cameras has proven to be a promising solution both in terms of privacy and security and energy efficiency.
The nature of the data acquired by these sensors, consisting of local variations in brightness, significantly reduces the risk of exposing sensitive personal information compared to traditional RGB video-based systems.
This approach gives greater importance to relevant movements, ignoring useless details such as colors, static subjects or background objects.
This feature makes them particularly suitable for applications where privacy protection is essential such as in the telemedicine field.
Furthermore, thanks to their asynchronous operating principle, a reduced energy consumption is also guaranteed.
This benefit is further amplified by the use of SNNs that use binary spikes, instead of complex multiplication operations, and exploit the concept of spatial and temporal sparsity, i.e. only a portion of the neurons is active at a given time.

7.3 Future Works

The multimodal nature of the dataset created and the gestures considered make it usable in numerous applications ranging from clinical research to the technology industry, as well as in gestural interfaces and robotic systems.

The dataset could in fact be extended to train models capable of distinguishing between healthy subjects and patients affected by the disease. Similarly, it can be used to develop home telemonitoring systems.
In the robotics field, neuromorphic sensors are particularly promising, since they allow efficient and low-energy processing. Our neuromorphic dataset could be used to train robots to recognize in real time the gestures performed by a human being and respond accordingly with a specific action.
In the research field, it can be used to train and optimize SNN or, more generally, as a benchmark to evaluate the performance of different models.

7.4 Conclusion

This thesis project contributed to the creation of a large multimodal dataset of hand gestures, commonly used in the assessment of neurodegenerative diseases. The data collection was made possible thanks to the use of different type of sensors, which provided different types of acquisition, from RGB frames to depth information, up to neuromorphic events. The integration of a motion capture system as a ground truth reference adds significant value, allowing accurate comparisons between different versions of the dataset in future research studies.
The neuromorphic dataset was pre-processed to maximize the quality and utility of the recordings, applying, after several tests, the transformations provided by the Tonic library. A custom class for interfacing with the dataset was also developed, specifically designed to facilitate its publication on Tonic and sharing with the scientific community, thus favoring further projects in the field of spiking neural networks and neuromorphic research.
To demonstrate a first practical use of the dataset, different Spiking Neural Networks (SNN) models were developed and a structured pipeline for hyperparameter optimization, training and cross-validation was implemented. The trained models were tested in the recognition of gestures present in the dataset, and the preliminary results, although improvable, are encouraging and offer useful indications for future iterations and improvements.

Bibliography

[1] Jimmy Phuong, Patricia Ordóñez, Jerry Cao, Mira Moukheiber, Lama Moukheiber, Anat Caspi, Bonnielin K. Swenor, David Kojo N. Naawu, and Jennifer Mankoff. «Telehealth and digital health innovations: A mixed landscape of access». In: *PLOS Digital Health* 2.12 (Dec. 15, 2023), e0000401. ISSN: 2767-3170. DOI: 10.1371/journal.pdig.0000401. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10723719/ (visited on 08/20/2024) (cit. on p. 1).

[2] Shilpa N. Gajarawala and Jessica N. Pelkowski. «Telehealth Benefits and Barriers». In: *The Journal for Nurse Practitioners* 17.2 (Feb. 2021), pp. 218–221. ISSN: 1555-4155. DOI: 10.1016/j.nurpra.2020.09.013. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7577680/ (visited on 10/07/2024) (cit. on pp. 1, 2).

[3] Marziye Hadian, Zahra Khakdel Jelodar, Mohammadreza Jabbari Khanbebin, Pezhman Atafimanesh, Ali Sarabi Asiabar, and Seyed Mehdi Hejazi Dehagani. «Challenges of Implementing Telemedicine Technology: A systematized Review». In: *International Journal of Preventive Medicine* 15 (Feb. 29, 2024), p. 8. ISSN: 2008-7802. DOI: 10.4103/ijpvm.ijpvm_48_23. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10982727/ (visited on 10/09/2024) (cit. on p. 2).

[4] Gianluca Amprimo, Giulia Masi, Gabriella Olmo, and Claudia Ferraris. «Deep Learning for hand tracking in Parkinson's Disease video-based assessment: Current and future perspectives». In: *Artificial Intelligence in Medicine* 154 (Aug. 1, 2024), p. 102914. ISSN: 0933-3657. DOI: 10.1016/j.artmed.2024.102914. URL: https://www.sciencedirect.com/science/article/pii/S0933365724001568 (visited on 08/20/2024) (cit. on p. 5).

[5] Foivos S. Kanellos et al. «Clinical Evaluation in Parkinson's Disease: Is the Golden Standard Shiny Enough?» In: *Sensors (Basel, Switzerland)* 23.8 (Apr. 7, 2023), p. 3807. ISSN: 1424-8220. DOI: 10.3390/s23083807. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10145765/ (visited on 09/26/2024) (cit. on pp. 5, 9).

[6] Jennifer G. Goldman et al. «Delivering Multidisciplinary Rehabilitation Care in Parkinson's Disease: An International Consensus Statement». In: *Journal of Parkinson's Disease* 14.1 (Jan. 23, 2024), p. 135. DOI: 10.3233/JPD-230117. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC10836578/ (visited on 10/23/2024) (cit. on pp. 8, 12).

[7] Jason K. Eshraghian, Max Ward, Emre Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. *Training Spiking Neural Networks Using Lessons From Deep Learning*. Aug. 13, 2023. arXiv: 2109.12894[cs]. URL: http://arxiv.org/abs/2109.12894 (visited on 04/10/2024) (cit. on pp. 20–22, 27–29).

[8] Guillermo Gallego et al. «Event-Based Vision: A Survey». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.1 (Jan. 2022). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 154–180. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2020.3008413. URL: https://ieeexplore.ieee.org/document/9138762/?arnumber=9138762 (visited on 10/25/2024) (cit. on p. 21).

[9] Riccardo Massa, Alberto Marchisio, Maurizio Martina, and Muhammad Shafique. *An Efficient Spiking Neural Network for Recognizing Gestures with a DVS Camera on the Loihi Neuromorphic Processor*. May 16, 2020 (cit. on p. 23).

[10] Nicolas Brunel and Mark van Rossum. «Quantitative investigations of electrical nerve excitation treated as polarization: Louis Lapicque 1907 · Translated by:» in: *Biological Cybernetics* 97 (Dec. 1, 2007), pp. 341–349. DOI: 10.1007/s00422-007-0189-6 (cit. on p. 27).

[11] Yuhuang Hu, Hongjie Liu, Michael Pfeiffer, and Tobi Delbruck. «DVS Benchmark Datasets for Object Tracking, Action Recognition, and Object Recognition». In: *Frontiers in Neuroscience* 10 (Aug. 31, 2016). DOI: 10.3389/fnins.2016.00405 (cit. on p. 33).

[12] Arnon Amir et al. «A Low Power, Fully Event-Based Gesture Recognition System». In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI: IEEE, July 2017, pp. 7388–7397. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.781. URL: https://ieeexplore.ieee.org/document/8100264/ (visited on 04/16/2024) (cit. on p. 34).

[13] Gianluca Amprimo, Claudia Ferraris, Giulia Masi, Giuseppe Pettiti, and Lorenzo Priano. «GMH-D: Combining Google MediaPipe and RGB-Depth Cameras for Hand Motor Skills Remote Assessment». In: *2022 IEEE International Conference on Digital Health (ICDH)*. 2022 IEEE International Conference on Digital Health (ICDH). July 2022, pp. 132–141. DOI: 10.1109/

ICDH55609.2022.00029. URL: https://ieeexplore.ieee.org/document/9861059 (visited on 08/16/2024) (cit. on p. 38).

[14] Joel S. Perlmutter. «Assessment of Parkinson Disease Manifestations». In: *Current protocols in neuroscience / editorial board, Jacqueline N. Crawley ... [et al.]* CHAPTER (Oct. 2009), Unit10.1. ISSN: 1934-8584. DOI: 10.1002/0471142301.ns1001s49. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2897716/ (visited on 09/05/2024) (cit. on p. 44).

[15] Gianluca Amprimo, Irene Rechichi, Claudia Ferraris, and Gabriella Olmo. «Objective Assessment of the Finger Tapping Task in Parkinson's Disease and Control Subjects using Azure Kinect and Machine Learning». In: *2023 IEEE 36th International Symposium on Computer-Based Medical Systems (CBMS)*. 2023 IEEE 36th International Symposium on Computer-Based Medical Systems (CBMS). ISSN: 2372-9198. June 2023, pp. 640–645. DOI: 10.1109/CBMS58004.2023.00293. URL: https://ieeexplore.ieee.org/document/10178872/?arnumber=10178872 (visited on 08/20/2024) (cit. on p. 44).