Alberto Sabater Bailón

Efficient scene understanding from video data

Director/es
Murillo Arnal, Ana Cristina
Montesano del Campo, Luis

ISSN 2254-7606

ISSN 2254-7606

Universidad Zaragoza
1542

Tesis Doctoral

EFFICIENT SCENE UNDERSTANDING FROM VIDEO DATA

Autor

Alberto Sabater Bailón

Director/es

Murillo Arnal, Ana Cristina
Montesano del Campo, Luis

UNIVERSIDAD DE ZARAGOZA
Escuela de Doctorado

Programa de Doctorado en Ingeniería de Sistemas e Informática

2023

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

ROBOTICS
UNIVERSIDAD DE ZARAGOZA

PhD Thesis

Efficient scene understanding from video data

Autor

Alberto Sabater Bailón

Directores

Ana Cristina Murillo Arnal

Luis Montesano del Campo

Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza, Abril 2023

Acknowledgments

There is no other way to start this writing but by thanking my thesis supervisors, these past four years of research would not have been the same without your continuous advice and support. *Ana C. Murillo*, thank you for your guidance, patience, commitment, and for keeping me motivated at all times. *Luis Montesano*, thank you for your valuable feedback and supervision, it has been essential for all we achieved in this thesis.

#TeamI3A, I feel deeply grateful to not only be working with exceptional researchers, but also with people that I can consider my friends. To the oldies and to the new joiners, thank you very much for all the conversations that helped to push the deadlines, for the brainstormings, and for that special eye you only have when it comes to making nice plots. Other than that, thank you for those never-ending coffee sessions, dramas, and wisdom, it will always remain written ~~in stone~~ with blue marker. Special thanks also to *Iñigo* and *Manu* for those coffees and conversations about Machine Learning, graphics, and internships, always insightful. *E. Montijano*, thank you for always taking *control* of the meetings, although *deep* inside you know that a bit of wildness always helps.

Thanks to all the people I met during my research stays, you all have made me feel at home. *Alex*, *Jose*, *Laura*, thank you very much for the warm welcome. Although a pandemic stood in the way, your support gave me a more than awesome stay in Lisbon. *Yasin*, *Richard*, *Pol*, and the rest of my Amazon team, you are the best buddies an intern can have, you made me feel an important and valuable member of the team. I will always miss those ping-pong and padel matches.

Although this manuscript summarizes the work of four years, this journey started a long time ago. The fact that I walked all this way along with so many friends makes me extremely proud and joyful. *Chipis*, thank you very much for the warm reception you gave me in those cold *Pilares*. Your kindness, joy, and special sense of humor have shaped the person I am now, and for that, I will always be indebted to you. Thanks to all my *friends from Bachelor*. Although we are not all living in the same city anymore, when we gather and grab a burger it feels like no time has passed. It feels awesome to have people you can always count on. Thanks to my favorite *AEGEE board*. Thank you for all the travels, memorable experiences, and *ruinas* we have shared. There is nothing better than sitting down together and looking back on our old battles and adventures.

Thank you to all my family. *Muchas gracias Mamá y Papá, gracias por todas las oportunidades que me habéis dado, por vuestra paciencia y comprensión, y por ser siempre un ejemplo de trabajo duro y resiliencia. Gracias hermana, por apuntar siempre a lo más alto, por siempre conseguir lo que te propones, por ser el mejor ejemplo que un hermano pequeño puede tener.* Finally, special thanks to *Agata*. Friend, partner in adventures, and family. Thank you for your unconditional kindness, care, and empathy.

It makes me very proud to write these lines, to look back and realize all the people who have walked by my side. In one way or another you all have helped me to get here. You are all part of this work, to the person I am, and to the person I will be tomorrow.

Abstract

Visual scene understanding is the process of extracting high-level information from visual data to gain a deeper understanding of the elements and entities in a scene, as well as to reason about their context and relationships. It is an essential area of research within Artificial Intelligence and Computer Vision and has applications in numerous fields, such as medical image analysis, autonomous robots or vehicles, and augmented and virtual reality. A special case of visual scene understanding is video data processing, which is often required in many real-world use cases. Video data processing, as opposed to still images, provides a more complete representation of the scene, but often presents specific additional challenges.

Neural Networks and Deep Learning have played a significant role in the processing of visual data, achieving state-of-the-art performance on many tasks. However, these methods have certain challenges and limitations, accentuated when processing video information instead of still images, which hinder their applicability for real-time or resource-constrained applications. Deep Neural Networks tend to be complex and computationally expensive, which often implies a high energy cost and latency. Moreover, Deep Learning algorithms typically demand a large amount of labeled data that is frequently difficult to obtain, and often struggle to generalize to new data domains. This thesis addresses some of these challenges and proposes different solutions for efficient video-based scene understanding, designed to learn from low-scale datasets and/or run with minimal computational resources. In particular, towards novel efficient scene understanding approaches, we work on improved video object detection and action recognition tasks, and the use of event cameras:

Object detection aims to localize and classify different objects in the scene. Although it has been widely studied for its application on still images, its performance on video data is more challenging. State-of-the-art video-based methods overcome specific video artifacts with complex and computationally expensive Deep Neural Networks. Differently, we propose a post-processing method that localizes possible temporal inconsistencies in the predictions of any object detector, and efficiently refines these detections with global information to better match the real objects.

Action recognition analyzes the human motion to identify the kind of action or gesture that is being performed. Real applications, like augmented or virtual reality, require the recognition of actions of variable nature, performed by different persons, and in heterogeneous environments. For this purpose, we propose two methods designed specifically for full-body and hand-only action recognition, based on the use of pose skeleton coordinates, that achieve these generalization capabilities.

Although RGB cameras are the most common sensors used for visual scene understanding, using *non-RGB sensors* can be beneficial for certain environments and applications. In this thesis, we study the use of *event cameras* due to their specific properties in scene representation and efficiency. These sensors capture only sparse illumination changes, ignoring the redundant static parts of the scene, and provide exceptional robustness to fast motions and challenging illumination conditions. Different from prior work, we effectively benefit from specific event data properties to achieve very high efficiency while also having a high performance in different scene understanding tasks.

All the code, trained models, and data developed in this thesis have been open-sourced for a broader impact on the scientific community and real-world applications.

Resumen

La comprensión visual de escenas es el proceso de extracción de información de alto nivel a partir de datos visuales para obtener un entendimiento más profundo de los elementos y entidades de una escena, así como para razonar sobre su contexto y relaciones. Es un área de investigación esencial dentro de la Inteligencia Artificial y la Visión por Computador y tiene aplicaciones en numerosos campos, como el análisis de imágenes médicas, los vehículos autónomos y la realidad aumentada y virtual. Un caso especial de comprensión visual de escenas es el procesamiento de datos de vídeo, que suele ser necesario en muchos casos. El procesamiento de datos de vídeo, a diferencia de las imágenes, proporciona una representación más completa de la escena, pero a menudo presenta desafíos adicionales.

Las redes neuronales y el aprendizaje profundo han desempeñado un papel importante en el procesamiento de datos visuales, logrando un alto rendimiento en muchas tareas. Sin embargo, estos métodos presentan ciertos retos y limitaciones, acentuados al procesar información de vídeo en lugar de imágenes fijas, que dificultan su aplicabilidad para aplicaciones en tiempo real o con recursos limitados. Las redes neuronales profundas tienden a ser complejas y costosas desde el punto de vista computacional, lo que a menudo implica un elevado coste energético y latencia. Además, los algoritmos de aprendizaje profundo suelen requerir una gran cantidad de datos etiquetados que a menudo son difíciles de obtener, y a pueden tener dificultades para generalizar a nuevos dominios de datos. Esta tesis aborda algunos de estos desafíos y propone diferentes soluciones para la comprensión eficiente de escenas basadas en vídeo, diseñadas para aprender de conjuntos de datos pequeños y/o ejecutarse con recursos computacionales mínimos. En particular, trabajamos en tareas detección de objetos de vídeo y reconocimiento de acciones, y en el uso de cámaras de eventos:

La *detección de objetos* tiene como objetivo localizar y clasificar diferentes objetos en la escena. Aunque se ha estudiado ampliamente para su aplicación en imágenes fijas, su rendimiento en datos de vídeo es más difícil. Los métodos más avanzados diseñados para procesamiento de video tratan artefactos específicos de vídeo con redes neuronales profundas complejas y costosas desde el punto de vista computacional. De forma diferente, nosotros proponemos un método de post-procesado que localiza posibles inconsistencias temporales en las predicciones de cualquier detector de objetos, y refina eficientemente estas detecciones con información global para ajustarse mejor a los objetos reales.

El *reconocimiento de acciones* analiza el movimiento humano para identificar el tipo de acción o gesto que se está realizando. Aplicaciones reales, como la realidad aumentada o virtual, requieren el reconocimiento de acciones de naturaleza variable, realizadas por diferentes personas, y en entornos heterogéneos. Para ello, proponemos dos métodos diseñados específicamente para el reconocimiento de acciones de cuerpo completo y de sólo manos, basados en el uso de coordenadas de poses, que consiguen estas capacidades de generalización.

Aunque las cámaras RGB son los sensores más utilizados para la comprensión visual de escenas, el uso de *sensores no RGB* puede ser beneficioso para determinados entornos y aplicaciones. En esta tesis, estudiamos el uso de *cámaras de eventos* debido a sus propiedades específicas en la representación de escenas y eficiencia. Estos sensores capturan sólo cambios de iluminación dispersos, ignorando las partes estáticas redundantes de la escena, y proporcionan una robustez excepcional frente a movimientos rápidos y condiciones de iluminación complicadas. A diferencia de trabajos anteriores, nos beneficiamos eficazmente de las propiedades específicas de los datos de eventos para lograr una eficiencia muy alta y, al mismo tiempo, un alto rendimiento en diferentes tareas de comprensión de escenas.

Todo el código, los modelos entrenados y los datos desarrollados en esta tesis son de código abierto para lograr un mayor impacto en la comunidad científica y en las aplicaciones del mundo real.

Contents

Chapter 1

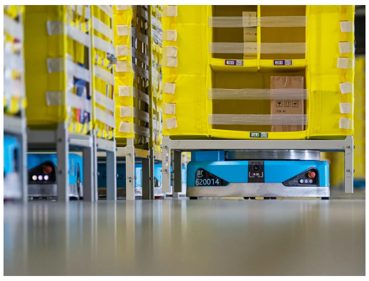
Introduction

Artificial Intelligence has experienced huge growth in recent years due to various factors. Firstly, the growing utilization of technological tools and connected devices provide an abundance of data for intelligent systems to learn from. For instance, in 2022, every minute 500 new hours of video were uploaded to YouTube, 66K photos were shared on Instagram, 1M hours were streamed, and 104 hours were spent in Zoom meetings[1]. Additionally, the development in computing technology and data storage capabilities makes it possible to process vast amounts of data quickly and efficiently. In a world that is becoming increasingly digital and where data production is continuously accelerating, we need to develop the right tools to effectively utilize that data for the creation of more intelligent and valuable applications.

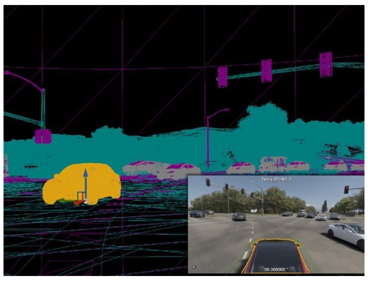
Visual data is a very important part of modern technology as it is increasingly prevalent in our daily lives. Within the field of artificial intelligence, computer vision focuses on developing algorithms and techniques to enable machines to interpret and understand this visual information of the world around us. One of the essential problems studied in computer vision is **visual scene understanding**, which leverages visual data to gain a deeper understanding of the elements and entities in a scene, as well as to reason about their context and relationships. These capabilities are important for a range of practical applications that have a high impact on our daily life. For instance, autonomous driving and smart surveillance benefit from the identification of persons and objects from a scene as well as their movement intention; Artificial or Virtual Reality (AR/VR) and human-machine interaction benefit from the identification of actions performed by persons and their interactions with objects; healthcare and medical assistance systems benefit from the automatic identification and segmentation of different anatomical structures or anomalies in medical images, which can aid in diagnosis and treatment planning; agriculture can benefit from visual crop analysis to optimize its growth, detect pests and diseases, and improve irrigation and fertilization practices. Figure 1.1 illustrates some of these examples.

Although some of these scene understanding tasks can be simplified to image data processing, many of them require or can be enhanced with the processing of video data instead of still images. **Video data** provides more complete information about what is happening in the scene over time, but it can also introduce more challenges for automated processing, such as motion blur, defocus, variable frame rate, occlusions, and atypical ob-

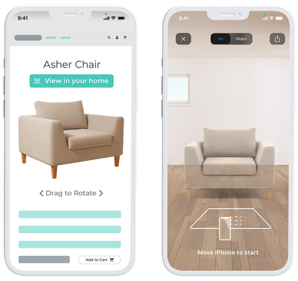
[1]`https://www.domo.com/data-never-sleeps`



(a) Factory automation in Amazon warehouses



(b) Autonomous driving at Tesla



(c) Augmented reality for interior design at Cylindo

Figure 1.1: Samples of applications that require visual scene understanding. *(Sources:* `https://www.aboutamazon.com`, `https://www.tesla.com`, `https://www.cylindo.com`*)*

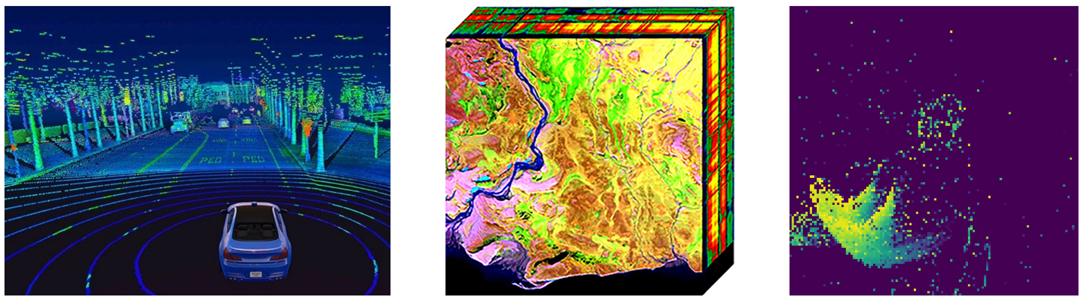
ject poses. To take advantage of the information provided by videos and address the new challenges they introduce, it is necessary to develop methods that take into account the temporal dimension. However, this often results in very complex solutions. For instance, specific video-based methods can overcome certain video artifacts but often come at the cost of heavy computational requirements, which increase their working latency and computational usage, making them less practical for real-world scenarios. Consequently, there is a need for **more efficient solutions** that can effectively process video data and improve the applicability of proposed solutions for real-world applications.

Besides video data, the use of **non-RGB recording cameras** can also be used to create a more complete representation of the scene. The use of different types of sensors, either individually or in combination with RGB cameras, can provide different information that benefits different scene understanding tasks. Of special relevance, depth cameras and LiDARs (see Fig. 1.2a) provide information about the distance and relative positions of objects in a scene, very useful for applications such as autonomous driving; infrared sensors can detect infrared radiation, making them ideal for autonomous surveillance in low-light conditions; hyperspectral cameras (see Fig. 1.2b) capture a wide range of the electromagnetic spectrum, making them suitable to monitor crops in agriculture; and event cameras (see Fig. 1.2c) capture fast and sparse changes in the scene, such as movement or changes in brightness, very valuable information for industry automation or autonomous driving. Despite the potential benefits of these non-RGB and multi-modal camera setups, research and development in these areas remain underrepresented in the scientific community, limiting their applicability in real-world scenarios.

Regardless of the data modality used, and similarly to many other fields in last years, recent advances in scene understanding are dominated by the use of deep learning. These models have the ability to learn intricate patterns from large datasets, achieving high levels of accuracy and generalization.

1.1 Deep Learning for Scene Understanding

As previously mentioned, the growth of artificial intelligence, and in particular the machine learning field, has been greatly accelerated by the increasing availability of data and advancements in computing technology. Unlike traditional methods which rely on human-



(a) LiDAR sensor (b) Hyperspectral image (c) Event camera [132]

Figure 1.2: Information recorded from different non-RGB sensors. *(Sources:* `https://velodynelidar.com/`, `http://rst.gsfc.nasa.gov/`*)*

defined rules, machine learning algorithms have the ability to learn from data, uncovering patterns that lead to more optimal solutions. A highly successful category of these algorithms is Neural Networks (NNs) [127, 169, 82] and Deep Neural Networks [81, 49] (DNNs), which excel even in the most difficult tasks. DNNs are constructed by stacking multiple and simpler Neural Networks (i.e., layers) that sequentially transform the input data. The final representation of the data encodes the information crucial for achieving the DNN's intended goal.

When it comes to computer vision, **Convolutional Neural Networks (CNNs)** [82, 75, 54] (see Fig. 1.3) are the most common Neural Network architectures. These networks are specifically designed to transform data with grid-like structures in one (e.g., audio samples) [157], two (e.g., images) [54], or multiple dimensions (e.g., videos) [66]. The core element of CNNs is the convolutional layer, which processes the input data by applying the so-called filters or kernels, that slide over the input features computing new output values at each step. Additionally, CNNs often incorporate Pooling layers that reduce the dimensionality of the input data, making use a more efficient use of computational resources. Although convolutions are a simple yet effective operation, they only capture short-range dependencies. Therefore, many convolutional layers must be sequentially stacked to learn wider spatial or temporal contexts.

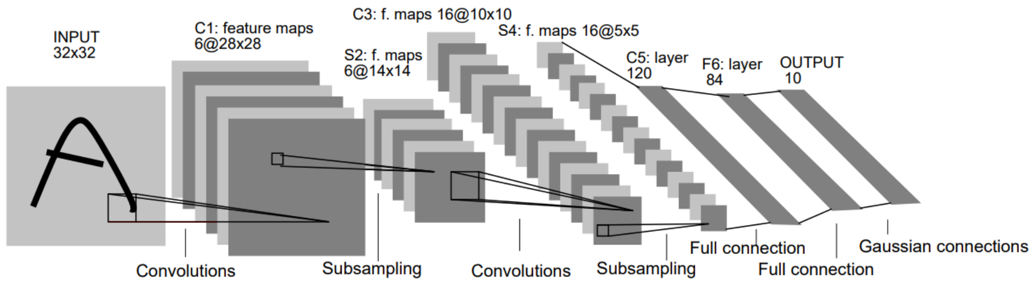


Figure 1.3: Scheme of a Convolutional Neural Network (LeNet-5 [83]). Convolutional filters slide over the input features generating a new data representation. Pooling layers reduce (subsample) the data dimensionality. Fully connected layers are used in the example to get the final prediction.

When it comes to the processing of data that has a temporal dimension, such as video data, **Recurrent Neural Networks (RNNs)** (see Fig. 1.4) are models of special relevance. Although there are many flavors of RNNs, including GRUs [29], LSTMs [56] and ConvLSTM [144], all of them share the feature of maintaining an internal state, also known as *memory*, that is utilized in the processing of input data at each time step and is modified to encode meaningful temporal information. This step can be repeated as many times as needed, allowing these models to theoretically handle data of any temporal length. However, due to their sequential structure, where the internal state is continually updated, RNNs find it difficult to capture long-term dependencies. Moreover, the learning phase is prone to suffer from exploding or vanishing gradients [12, 116] and, since the data processing is not parallelizable, they are slow to train. Despite these limitations, RNNs are a frequent choice for language [78, 152], audio [16, 176], or video data [156, 103] processing.

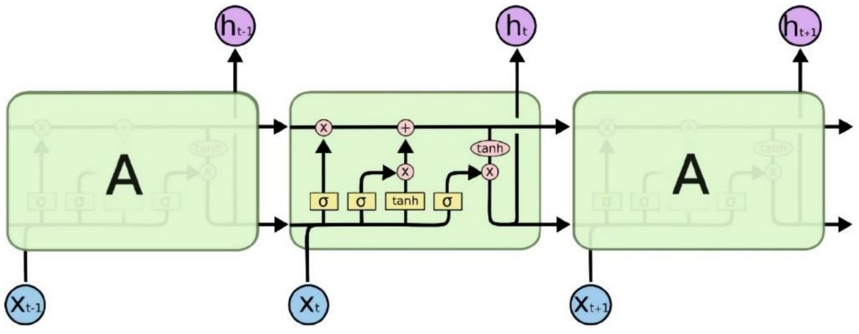
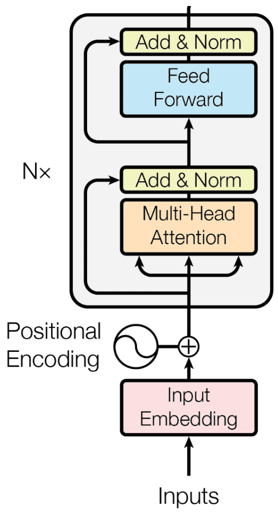


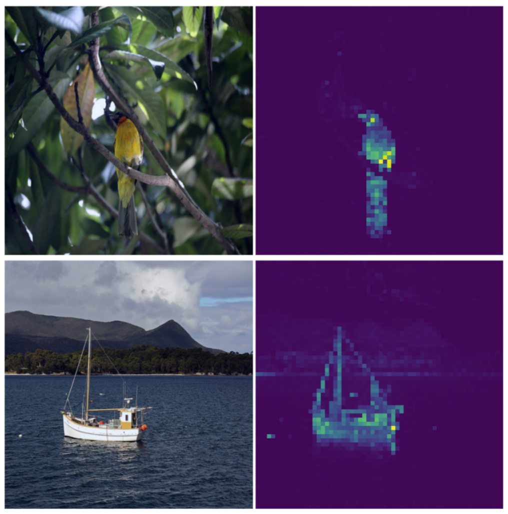
Figure 1.4: Scheme of a Recurrent Layer (LSTM). Layer output h_t at time t not only depends on the layer input x_t, but also on the context given from the previous time step. *(Source: `http://colah.github.io/`)*

Another significant architecture in computer vision is the attention-based models or **Transformers**. These models (see Fig. 1.5a) break down the input data, of arbitrary length, into smaller tokens that are then transformed with attention mechanisms. This attention, *Multi-Head Self Attention*, is able to capture both short and long-range dependencies (see Fig. 1.5b) among the input tokens, which allows giving different importance to different parts of the input data. Different from CNNs, Transformers do not have an inductive bias, so they make use of positional information to provide the input tokens with additional information, such as spatial or temporal information. Although they were initially designed to process temporal information such as language data [160, 70], in recent years they have gained a lot of popularity in other areas such as visual perception [22, 36, 96, 97], audio processing [119, 86], or multi-modal learning [118, 3, 2]. Despite their high performance, base Transformer implementations imply a high computational cost that hinders their scalability and efficiency in resource-constrained scenarios.

In visual scene understanding, simpler methods often process still images with CNNs of variable complexity to perform tasks such as object detection [123, 48], semantic segmentation [53, 4], or pose estimation [20, 107]. However, when it comes to the processing of video data, deep learning solutions need to make use of one or more of the above-mentioned models to handle the temporal context, such as LSTMs [156, 180, 110, 100, 156], ConvLSTMs [177, 149], attention layers [97, 2, 7], or jointly processing stacked frames with 3D-Convolutions [66, 155]. Despite the capabilities of these methods, they tend to have complex architectures that require high computational resources, making them unsuitable for real-time or resource-limited environments. Moreover, these complex architectures of-



(a) Transformer architecture [160]



(b) Attention maps from different heads [23]

Figure 1.5: a) Shows the base Transformer architecture, where the input tokens are processed with Multi-head Self Attention layers. b) Shows a visualization of different attention maps, where each head (each color) has learned to recognize different long and short-range dependencies from the data.

ten require more data to be trained, which can be a challenge for supervised methods that rely on labeled data. Therefore, there is a need for the creation of models that are more efficient in terms of computational complexity and training data requirements.

1.2 Efficiency in Deep Learning

With advancements in computer capabilities, deep learning models have become increasingly complex and are being used for more intricate tasks. These complex models require more and more data and energy up to the point that efficiency, both during training and when deployed, has become an important factor in research as well as in practice.

The recent trend consists of using vast amounts of unlabeled data scrapped from the Internet [139, 120] to train **large *Foundation Models*** with unsupervised, self-supervised or weakly-supervised learning [17, 23, 118], to later be used or fine-tuned for different downstream tasks. All of these top-performing methods grow in the number of parameters (see Fig. 1.6), which in turn results in longer training times, **higher computational requirements**, and higher latency. Although these models achieve better performance than more efficient solutions, they are not always ideal for certain applications that require real-time processing or are resource-constrained, such as autonomous driving; or those that additionally must run on edge devices with limited computational power, such as AR/VR.

As a consequence of the increased computational requirements, an important drawback of large deep learning models that is becoming increasingly relevant is their **ecological footprint**. The training and use of large models require significant energy resources from the data processing and machine refrigeration which can result, depending on the energy source, in significant CO_2 emissions. As a reference, it is estimated [101] that Foundation

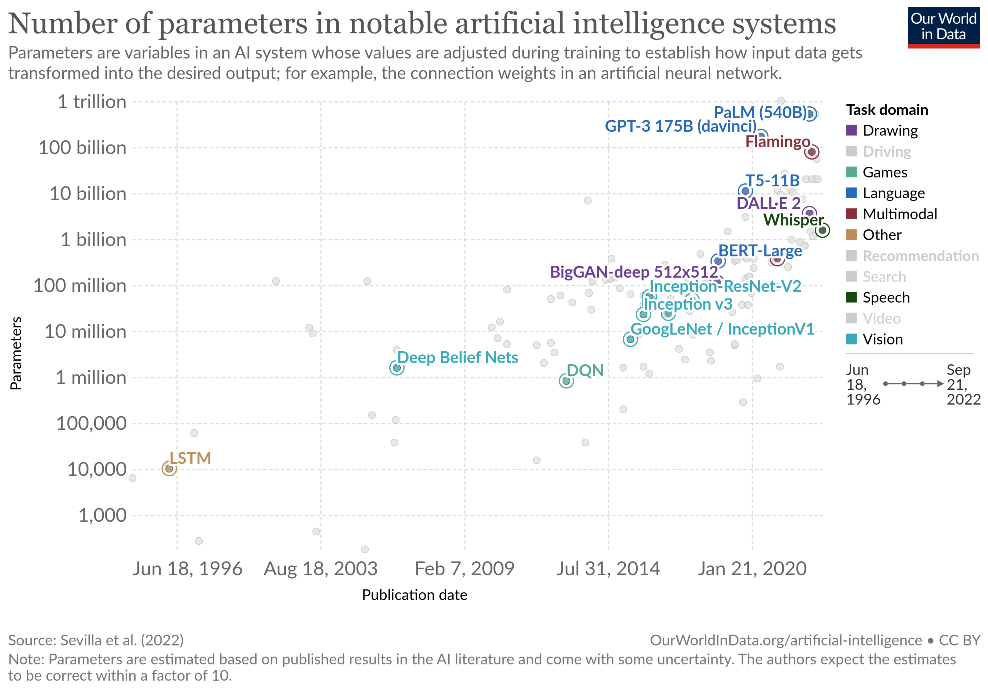


Figure 1.6: Evolution of the number of parameters of top-performance deep learning models in different task domains. *(Source: `https://ourworldindata.org/`)*

Models such as BLOOM [137] and GPT-3 [17] required 433 MWh and 1,287 MWh of energy respectively only for their training, which is equivalent to the electricity consumed over a year in 36.4 and 108 homes respectively (reference data[2] from USA homes in 2019). Moreover, the increasing demand for computing power means that older technological devices are becoming obsolete, leading to a set of materials that are difficult to recycle or repurpose.

In addition, these large deep learning models often require extensive and diverse datasets for their training, which can be challenging to obtain and time-consuming to label accurately. Moreover, once a model has been trained on a specific learning setup, its **generalization different data domains** may cause a drop in performance [115], making it challenging to use in real-world applications. For instance, models trained with data from automovilisttic simulators might have lower performance in real driving scenarios, and action recognition models trained with first-person view recordings might have a performance drop when the action recordings have an external perspective. Additionally, real applications might require the recognition of elements not seen during training. In these cases, deep learning models must be able to learn good data representation that eases their later evaluation or be able to learn from a few data samples or low-scale datasets [147, 148].

In parallel to the model design, using the **right data formatting** can ease the data processing to build more efficient deep learning models or to achieve a better generalization. An example is the case of action recognition, which can be performed by analyzing RGB video sequences [35, 182], but also by processing human skeleton coordinates [179, 98]. The latter presents a much lower dimensionality than images and abstracts the person's appearance, easing the generalization for action recognition in other individuals. Another example

[2]`https://www.epa.gov/energy/greenhouse-gas-equivalencies-calculator`

is the use of event cameras [15, 61], instead of the traditional RGB ones. These sensors just log sparse changes in the scene, so the latter data processing avoids the processing of redundant video information, as it happens when RGB frames or skeleton coordinates barely change in time.

Therefore, it is necessary to explore alternative methods that offer efficient and responsible use of computational resources, while achieving also high performance. These methods should be able to run on edge resource-limited devices and provide real-time feedback. Moreover, deep learning solutions should be able to generalize to similar scenarios or domains without requiring extensive effort from end users. This thesis addresses all of these requirements in order to advance the field of visual scene understanding toward more practical and ecological solutions.

1.3 Challenges and Contributions

This doctoral thesis studies and develops novel deep learning approaches for scene understanding, based on video data processing, and with a strong focus on their efficiency and applicability to real-world use cases. For this purpose, the presented research is focused on three main questions or challenges:

Can we develop more efficient solutions?
As previously described, deep learning models often require heavy computations to work, especially when processing video data, which directly affects the prediction time and energy consumption. However, real-world scenarios usually require real-time processing but they are constrained by limited computational capabilities and battery power. Our proposed research tackles this challenge by developing highly efficient deep learning algorithms that can provide real-time predictions while maintaining strong performance. By using less computationally intensive techniques, these models are often even able to work in CPU instead of GPU, enabling their use in a wider range of real-world scenarios.

Can we learn good models with low-size datasets?
Training deep learning models often requires the use of large labeled datasets that are often not available and whose creation is not always feasible. Additionally, once the models are trained, they are constrained to predict the same labels they saw during training and struggle to generalize to different data domains. In order to mitigate these issues, our work is adapted to learning with low-scale datasets when required, and includes techniques such as contrastive learning or N-shot prediction, aimed to increase the generalization to unseen data domains and to predict non-learned labels.

Can we benefit from non-RGB sensors?
RGB cameras are widely used in scene understanding, but other data modalities and recording sensors also provide valuable information for certain tasks. Our proposed research not only includes RGB data processing but also explores the use of other modalities. In particular, we use human skeleton coordinates to model human motion, and data recorded with event cameras for its potential to improve the efficiency in scene understanding and as a complement to RGB recordings to improve the model performance.

Among the multiple scene understanding topics, our research covers two essential tasks in video processing, object detection and action recognition, and analyzes the use of event cameras, whose efficiency and performance have a direct impact on real use cases. In the following, we introduce each of these topics, the main methods used to address them, and their main challenges. Then, we motivate and introduce our proposed methods.

1.3.1 Object Detection

Object detection stands for the problem of recognizing and localizing different objects in the scene. As observed in Figure 1.7, the object detection output is a set of bounding boxes defined by their coordinates, a class label, and its confidence score.

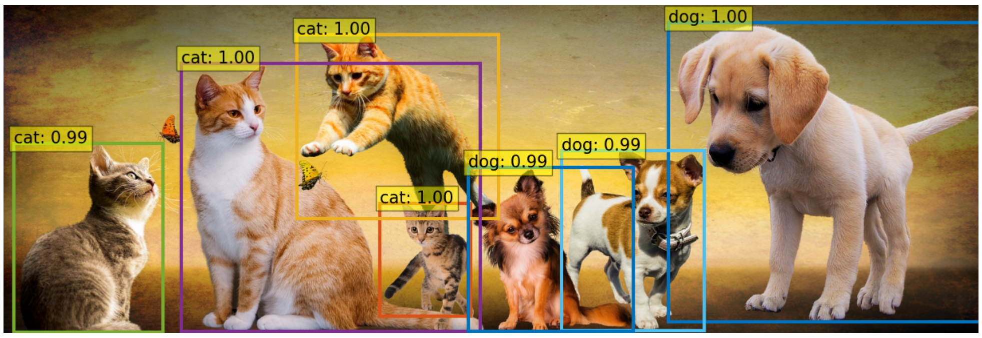
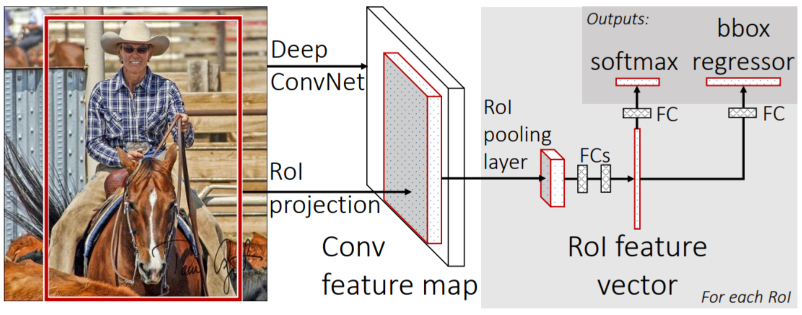


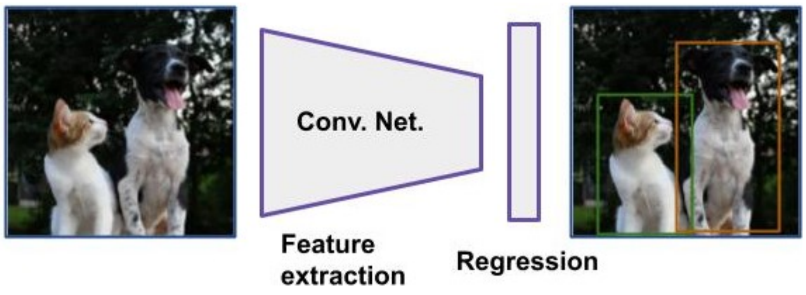
Figure 1.7: Object Detection overview

Object detection solutions, depending on their design, can work either in one or two stages. Two-stage detectors (see Fig. 1.8a) like R-CNN [48] or Mask R-CNN [53] consist of a first step that generates region proposals that might contain an object, and a second stage that processes these regions to regress the class label and confidence score. Differently, one-stage detectors (see Fig. 1.8b) like SSD [95] or YOLO [123] predict bounding boxes and class scores in a single forward pass. Consequently, one-stage detectors have a higher efficiency than two-stage ones, being able to perform even in real-time, but they suffer from a worse performance.

Although object detection research is often simplified to learning from still images, we find numerous real use cases such as autonomous driving, surveillance, or AR/VR, that require the processing of video data. Video object detection presents extra challenges since the data is usually not as clean as when working with still images. Video artifacts like defocus, occlusions, motion blur, or uncommon object poses make the elements in the scene more difficult to recognize. For example, an object that is moving fast in the scene can present different levels of motion blur or perspectives in consecutive frames, making its recognition uneven across frames. For this purpose, specific video object detectors tackle this problem by leveraging the temporal information and analyzing wider spatio-temporal information. This is performed by sharing or propagating frame features generated at different steps of the video. For this purpose, 3D-DETNet [87] uses 3D-Convolutions, [177] uses ConvLSTM layers, and [100] uses LSTMs for this purpose. Although these architectures perform better than still image object detectors, their more complex architectures lead to lower efficiency, hindering their use for real applications.



(a) Two-stage Object Detector (scheme of Fast RCNN [47])



(b) One-stage Object Detector (scheme of YOLO [123])

Figure 1.8: Different modalities of Object Detection architectures

Different from these solutions, we tackle the problem of video object detection with a fast post-processing step that can be performed over the detection results of any image or video object detector [131]. By analyzing location, geometry, appearance, and semantic information, we are able to link predicted objects across time to refine their class scores and coordinates with more global information. Refined object detections better match the objects they refer to, including objects that were not detected in the first instance, objects that were miss-classified, and those whose predicted bounding box was not properly fitting the real object. Finally, the use of this post-processing method makes the performance of efficient image object detectors get closer to more complex video object detectors, but also boosts the state-of-the-art performance of the most robust video object detectors.

1.3.2 Action Recognition

Action recognition (see Fig. 1.9) is the problem of recognizing the type of activity that a person is performing, typically from a video sequence. Solutions to this task have numerous applications such as human-computer interaction, medical rehabilitation tasks, or AR/VR.

However, different from object detection, action recognition datasets available in the research community are not as large and heterogeneous. Therefore, the set of actions that can be learned and recognized is very narrow, and generalization from training data to real environments is difficult. These issues have pushed the research of action recognition models in two directions: the use of data that abstracts the appearance of a person to focus solely on its motion, and the recognition of actions non seen during training.

Regarding the data used for the recognition, initial models used the raw RGB video data [35, 182], which hinders its generalization and applicability to other video scenes. In order to abstract the motion from the appearance, other methods were based on the

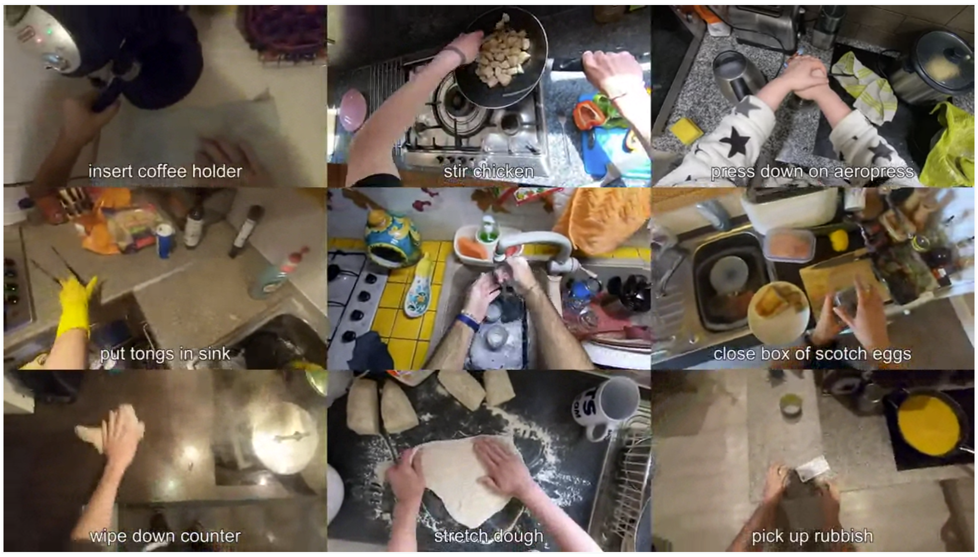


Figure 1.9: Action Recognition. First-person view actions from the EPIC-KITCHENS-100 dataset [31]

processing of depth maps [113], or body skeletons [179, 98]. The latter one has become a standard for action recognition due to the development of methods [21, 20] able to extract skeleton coordinates in real-time (often integrated into camera sensors [183]), and due to its easier and faster processing.

Regarding the recognition of actions not seen during training, a line of research [91, 92, 51, 65, 102] studies the creation of descriptors that summarize a certain motion. Then, given the descriptors from one or few reference actions not seen during training (N-shot prediction), we can recognize them on new video sequences.

This thesis contributes with novel methods for the recognition of actions performed both with the full body [134] or just with the hands [130]. In both cases, we push the model generalization to unseen scenes by working with human skeleton data, abstracting the recording camera viewpoint, and performing N-shot evaluation.

Skeleton-based full-body action recognition. Full-body actions are the ones performed with the whole body, such as jumping, running, or kicking. In our case, we represent the human body with 3D skeleton coordinates, that are relative to the camera location. The latter limits the learning and hinders the action recognition performance in setups with different camera perspectives, and when the recorded person is in a location or moves towards a location not represented in the training data. In order to overcome these limitations, we engineer a new viewpoint invariant pose representation, based on geometric features extracted from skeleton coordinates.

Given the video sequences of viewpoint invariant pose representations, we generate human motion descriptors and, with an N-shot approach, we are able to detect and recognize actions in other video sequences. Our detection pipeline specifically introduces a set of improvements to achieve more accurate action recognition in the most challenging scenarios. In particular, we evaluate our action recognition framework in a real-use case of therapies with autistic people. This environment consists of action imitation games with high-level artifacts in the patient's motion, whose analysis provides meaningful qualitative and quantitative information to the therapist, essential to evaluate the attention and coordination of the patient.

Skeleton-based hand action recognition. Hand action recognition aims to recognize the actions or gestures, such as sliding or grasping gestures, that are only defined by the movement of the hands. In this case, only the skeleton coordinates related to the hands are required for the recognition task. However, different from full-body actions, hand gestures depend on a point of reference for their definition. E.g., by looking just at the hand coordinates, the gesture of *pointing to the left* can be recognized as *pointing to the right* (or any other direction) depending on the camera viewpoint. Therefore the camera viewpoint invariance introduced in our previous work is no longer valid.

Our work for hand action recognition looks for a robust generalization to different camera viewpoints (e.g., training with an egocentric view and evaluating with a frontal perspective) and action domains (e.g., training with human-robot interaction gestures and evaluating human-object interaction actions). For this purpose, we use an optimized and simplified hand pose representation and a Neural Network trained with randomly simulated camera perspectives. For the evaluation, our framework is able to generalize to any camera perspective with N-shot evaluation and to recognize actions unseen during training.

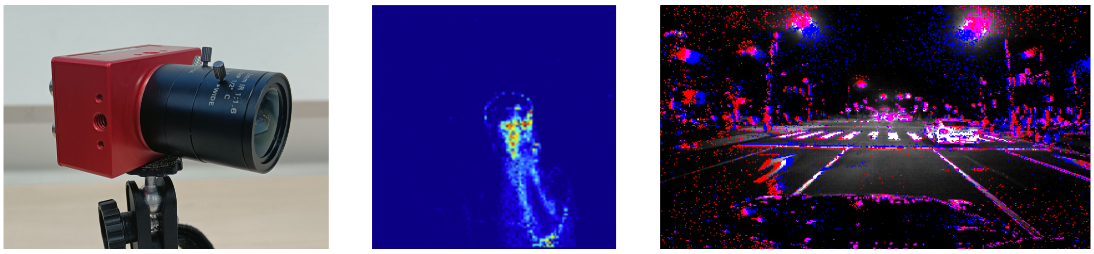
1.3.3 Scene Understanding with event cameras

When it comes to scene understanding and computer vision, RGB cameras are the most commonly used sensors due to their low cost and availability in our daily life, e.g., their presence in smartphones or home appliances. These sensors can capture very distant information and record appearance information such as color and texture with high fidelity. However, their performance is limited in certain scenarios, such as extreme lighting conditions, which can result in inaccurate scene representation, or fast-moving objects, which appear blurred and of low quality. These limitations have significant implications in cutting-edge applications such as autonomous driving and fast action recognition, where high accuracy and resolution are essential.

This thesis tackles these issues by using event cameras. Different from traditional cameras, event cameras (see Fig. 1.10a) just register sparse illumination changes (*events*), which in practice correspond to the motion happening in the scene and ignores the static information that does not provide extra meaningful information. These sensors also have minimal power consumption and present a high temporal resolution and high dynamic range, making them very robust to challenging lighting conditions and fast motions. The characteristics make event cameras suitable for applications such as action recognition [15, 61], depth estimation [45, 167], or odometry [72, 126], especially in environments with low computational capabilities.

Although these cameras have a high potential, computer vision solutions that process their information are not mature yet. On the one hand, we find efficient solutions that model the events as sparse representations like point clouds [140, 161] or graphs [165, 14, 15, 34] that can be processed very fast, but do not achieve high performance. On the other hand, we find solutions that create dense frame representations (as observed in Fig. 1.10b and 1.10c) from events, using different techniques [77, 108, 46, 61], that are processed with

CNN-based Neural Networks. The latter achieves higher performance but at a higher computational cost, since it ignores characteristics of these sensors such as their sparsity and high temporality.



(a) Event Camera (Davis 346) (b) SL-Animals-DVS Dataset [158] (c) MVSEC Dataset [187]

Figure 1.10: Event cameras. a) An event camera sensor. b) A sample frame representation of events from an action recognition dataset that shows a person moving the arm. c) A sample frame representation of events from a depth estimation dataset. As observed, b) and c) only register information on the parts of the scene that present some motion or changes in their texture. Remaining visual information is empty.

The present thesis introduces a different way of processing event information, designed to have high performance while being efficient[133, 132]. This new framework introduces a new event representation that benefits from the robustness of dense frame representations, but also from the event data sparsity by ignoring the parts of these frames that do not contain relevant information. This new event representation is then processed by an attention-based Neural Network that naturally tackles the sparsity of this data to perform efficiently while achieving high performance.

The proposed event processing framework is tested in different public datasets of event-stream classification, in particular, human action recognition. These sequences include motion of different complexity and with challenging lighting conditions. Final results suppose a boost of accuracy and efficiency with respect to previous methods, being able to perform online inference both in GPU and CPU. Moreover, we probe that this new framework can be adapted to other computer vision tasks with event data. In particular, we adapt it to predict dense labels, i.e., depth maps from driving sequences. Results show how, even though dense heads do not benefit that much from the event sparsity, our proposed framework is still able, by using different mechanisms, to achieve high precision and still be able to perform in real-time.

1.4 Summary of Results

The results of all the work developed during this thesis have been published in different top international conferences and journals; and the related code, data, and trained models have been open-sourced in GitHub repositories.

A. Sabater, L. Montesano, A. C. Murillo. **Robust and efficient post-processing for video object detection**. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS 2020. *Repository:* `https://github.com/Alberto`

Sabater/Robust-and-efficient-post-processing-for-video-object-detection

A. Sabater, L. Santos, J. Santos-Victor, A. Bernardino, L. Montesano, A. C. Murillo. **One-shot action recognition towards novel assistive therapies**. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. Learning from Limited and Imperfect Data. CVPRW 2021. *Repository: https://github.com/AlbertoSabater/Skeleton-based-One-shot-Action-Recognition*

A. Sabater, I. Alonso, L. Montesano, A. C. Murillo. **Domain and view-point agnostic hand action recognition**. IEEE Robotics and Automation Letters. RA-L 2021. Presented in IROS 2021*Repository: https://github.com/AlbertoSabater/Domain-and-View-point-Agnostic-Hand-Action-Recognition*

A. Sabater, L. Montesano, A. C. Murillo. **Event Transformer. A sparse-aware solution for efficient event data processing**. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. Efficient Deep Learning for Computer Vision. CVPRW 2022. *Repository: https://github.com/AlbertoSabater/EventTransformer*

A. Sabater, I. Alonso, L. Montesano, A. C. Murillo. **Event Transformer^{+}. A multi-purpose solution for efficient event data processing**. Under review. 2022. *Repository: https://github.com/AlbertoSabater/EventTransformerPlus*

This work has been supported by FEDER/Ministerio de Ciencia, Innovación y Universidades – Agencia Estatal de Investigación projects FILOVI (RTC-2017-6421-7), PGC2018-098817-A-I00 and PID2021-125514NB-I00, DGA T45 20R/FSE and the Office of Naval Research Global project ONRG-NICOP-N62909-19-1-2027.

During this thesis, I also had the chance to do two **research internships at different research entities**. First, I did a 4-month visit at the *VisLab Group*, at *ISR-Lisboa* (*Instituto Superior Tecnico*, Lisboa), which resulted in the publication of *One-shot action recognition towards novel assistive therapies.* Second, I interned for 4 months at *Amazon-Berlin* as *Applied Scientist*, where I researched on multi-modal learning (images and multi-lingual text) and large-scale Neural Networks.

In this period I also collaborated in the **supervision of the Master Thesis** *Graph Neural Networks for Corner Detection with Event Cameras* of Adrian Schneebeli from the Master *Robotics, Systems, and Control* in *ETH Zürich* (Switzerland). And I collaborated as **Teaching Assistant** in the *Master in Robotics, Graphics, and Computer Vision* from the University of Zaragoza (15h).

Along with the presented publications, I also collaborated as a **reviewer for different journals and conferences**: *RA-L (2020, 2021, 2022, 2023), CVPR (2021, 2022, 2023), ICRA (2020, 2023), IROS (2020, 2021).*

Finally, I worked as a **volunteer** in the *Emerging Technologies and Factory Automation Conference* (ETFA 2019) and *Iberian Robotics Conference* (ROBOT 2022)

1.5 Manuscript Organization

The following chapters describe the four main contributions to the field of computer vision and scene understanding introduced above. Chapter 2 introduces our post-processing method for video object detection. Chapter 3 presents our skeleton-based action recognition model robust to challenging motion conditions. Chapter 4 introduces our skeleton-based hand action recognition model generalizable to variable recording perspectives and hand gestures. Chapter 5 describes our new approach to efficiently process data recorded from event cameras. Each one of these chapters includes its own specific related work discussion as well as the corresponding conclusions. Finally, Chapter 6, summarizes the conclusions of the presented thesis and develops our vision for future work.

Chapter 2

Efficient post-processing for video object detection

As mentioned in the introductory Chapter, video data presents unique challenges that hinder the effectiveness and applicability of the object detection methods originally designed for still image processing. On the other hand, specific video-based solutions tackle these video artifacts and achieve state-of-the-art performance by leveraging the temporal consistency of video data. However, they require computationally-intensive deep learning techniques, not suitable for resource-limited applications. Differently, we demonstrate how we can also benefit from this temporal consistency, not only in the object detection model but also by analyzing the temporal consistency of the predicted object detections across frames. In the following, we present a post-processing method that refines the object detections in an efficient manner to better fit the real objects. The simplicity of this method makes it robust to different video artifacts, such as fast object motions, and is well-suited for use in low frame rate videos.[1]

2.1 Introduction

Many application fields such as robotics, surveillance, or wearable devices require object detection over their embedded camera video streams and efficient algorithms to process them. Deep learning-based detection approaches, such as the well-known YOLO [123] or MaskRCNN [53], have boosted image object detection performance in recent years. However, there is still a large gap between object detection performance in video and images, mainly because video data is more challenging due to numerous artifacts and difficulties such as blur, occlusions, or rare object poses.

Two main strategies have been explored to improve object detection in videos. On one hand, there are detection models specifically designed to work over video streams [190, 171]. They typically implement feature aggregation from nearby frames and achieve higher accuracy than still image detectors, but they are often slow and require heavy computation. This makes them not well suited for applications in low-resource environments, like wearable devices, or for applications where near real-time computing is a requirement, such as

[1]Code and learned models available at: `https://github.com/AlbertoSabater/Robust-and-efficient-post-processing-for-video-object-detection`

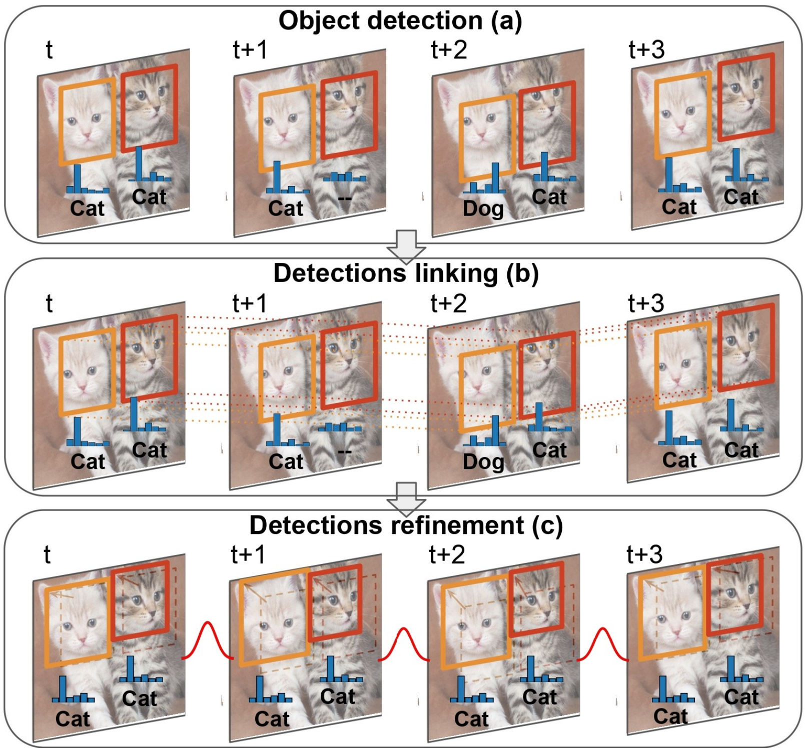


Figure 2.1: Proposed post-processing pipeline to improve video object detection. A standard image detector predicts object instances in a sequence of video frames (a). Our approach links object instances across frames based on the learned similarity evaluation (b) and uses contextual information to refine the detections (c), both object classification and location.

robotics or monitoring video analysis. On the other hand, post-processing methods such as Seq-NMS [52] and Seq-Bbox-Matching [11] have been proposed to process the outputs of an image object detector evaluated on the video frames to improve the performance. They are mostly based on linking the predicted objects across frames and using these links to refine the detection results. This strategy is typically much faster than specific video object detection methods.

The key to post-processing methods is the way they relate detected objects among consecutive frames. This linking is usually based on hand-made heuristics, a common one uses the Intersection over Union (IoU) between object detections. This approach has several limitations. First, the base detector does not always predict reliable bounding box coordinates. Second, IoU values strongly depend on displacements due to camera or object motion. For instance, fast-moving objects may not present enough overlap to get a reliable linking. Note that the same effect occurs when the frame rate drops (e.g., due to computational constraints). Finally, the presence of multiple objects simultaneously in the scene also makes heuristics difficult to design and prone to failure.

This paper presents a novel post-processing pipeline for video object detection (see Fig. 2.1) that can be used in conjunction with any video or image detector. The main novelty relies on the way the similarity is evaluated between object detections to link them across frames. We propose to use a learning-based similarity function that combines different descriptors and is designed to be more robust to varying speeds of object motions. Once all possible object instances are linked across frames, a refinement step is run to improve both the classification and location of object detections.

We evaluate our method against state-of-the-art post-processing methods for image detectors on the well-known video dataset ImageNet VID [129], obtaining better results mainly due to more robust links for fast-moving objects. We also show that our method improves the performance of specific video object detectors. Interestingly, the increased robustness to fast-moving objects implies we can process frames more sparsely, and then allows us to use more computationally demanding object detectors even if time constraints are high and not all frames can be processed.

2.2 Related Work

Object detection in videos often builds on top of image object detection. The latter is a well-studied problem with current state-of-the-art methods based on deep learning architectures. Multi-stage detectors [125, 30] follow R-CNN [48] and split the prediction process into two stages: candidate selection and candidate classification. Single-shot models, on the other hand, use a single Neural Network trained end-to-end to perform object detection in a single step. Many variants exist [95, 123, 80, 186, 185] with different object representations. They are in general faster but perform worse than multi-stage ones.

There are two types of approaches to extend object detection to video and cope with its specific challenges (blur, occlusions, rare poses) and to exploit the temporal information and consistency in video data:

Video Detectors. Video object detectors are designed to exploit the surrounding context of a frame and usually, the model propagates or shares object features across frames. FGFA [190] aggregates nearby features along the motion path given by an optical flow estimation. D&T [40] trains a ConvNet end-to-end both for object detection and tracking by using correlation across feature maps and re-scores linked detections. SELSA [171] extracts object proposals from different frames of a video and aggregates their features depending on their semantic similarities. TSM [89] shifts features along the temporal dimension to perform object detection with 2D CNNs. TCD [184] conditions the output of a single image detector by the tracklets calculated in previous steps.

Video object detectors are usually more computationally expensive than detectors working on still images due to the increased network complexity, their need to process more data, and often the requirement to calculate additional data such as optical flow.

Post-processing methods to improve video detection. Post-processing methods incorporate temporal context information into the output predictions of either image or video object detectors. Applied to per-frame detections, these methods speed up the inference pipeline with respect to specific video detectors while boosting the final detection performance with respect to their base detector.

Some post-processing methods are based on Kalman Filter variations and use tracking ideas to make the detections more robust or consistent. These are often applied to specific domains like person re-identification, where persons have to be detected and tracked in videos [106, 24] extracted from cameras usually with a fixed position. Deep SORT [170] improves the SORT [13] algorithm, based on Kalman Filters and the Hungarian Algorithm, by adding appearance information to each predicted bounding box.

Other strategies applied in more general settings are based on bounding box propagation or matching across frames. For example, T-CNN [68] uses context information to suppress false positives and optical flow to propagate detections across frames to reduce false negatives. Seq-NMS [52] matches high overlapping detections across frames within the same clip to make detection results more robust. Seq-Bbox-Matching [11] links overlapping detections from continuous pairs of frames to create and re-score object instances and uses them to infer missed detections.

Our proposed approach is related to this last group of post-processing techniques. We perform bounding box linking across frames, but instead of hand-made heuristics, a learned classifier is used to distinguish whether two detections belong to the same object instance or not. This model exploits both intermediate features from the base object detector and additional properties of the bounding box.

2.3 Proposed Framework

Our proposed approach for video object detection runs the three modules summarized in Fig. 2.1, which are detailed next.

2.3.1 Object detection and description

Our approach works on top of any initial object detector that can provide object bounding boxes and class confidence score vectors. For each video frame t, we get a set of object detections, and each object detection o_t^i is described by:

- **Location and geometry**, i.e., its bounding box information: $bb_t^i = \{x, y, w, h\}$.
- **Semantic information**, i.e., the vector of class confidences cc_t^i provided by the detector. $cc_t^i \epsilon \mathbb{R}^C$, where C is the number of classes within the dataset.
- **Appearance**, i.e., a L2-normalized embedding, $app_t^i \epsilon \mathbb{R}^{256}$, representing the appearance of the patch. It is learned as detailed next.

Figure 2.2 summarizes how these descriptors are obtained. The first two are directly provided by the base object detection model. The appearance descriptor is computed from a set of feature maps generated by an intermediate layer of the base model. We propose a simple architecture to learn this appearance embedding as shown in the figure. A RoI Pooling Layer [125] is used to extract the feature map outputs that correspond to each of the predicted bounding boxes and scale them to fit a pre-defined shape. Since such intermediate descriptors are generally large, we use a single Fully Connected layer to learn a mapping to a lower dimensional embedding and limit the memory and computational resources. This embedding model is trained to minimize the triplet loss proposed in [138]:

$$\sum_i^N \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|]f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+ , \tag{2.1}$$

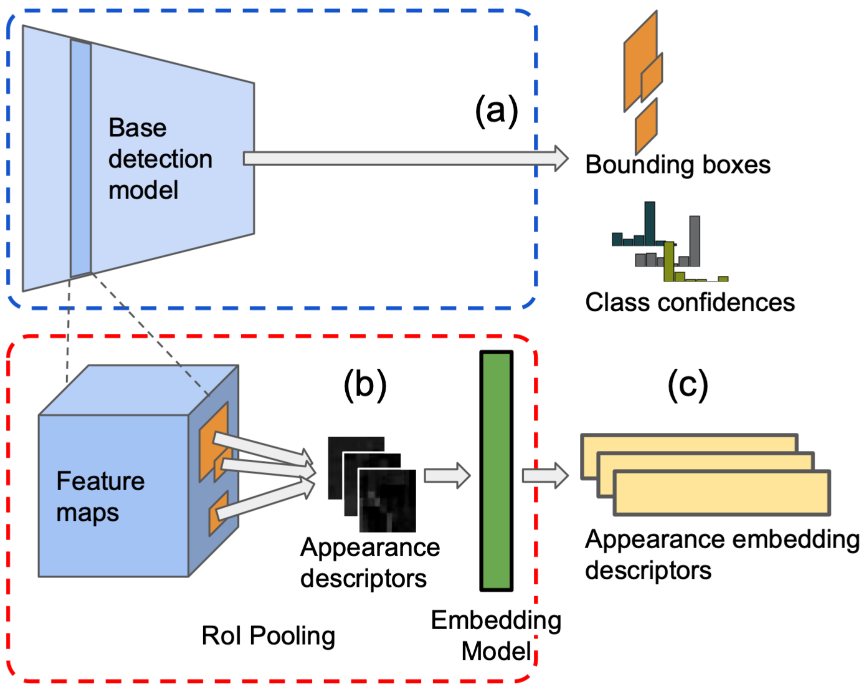


Figure 2.2: **Object detection and description**. Each detection from the base object detector is represented by the bounding box and vector of class confidences provided by the base model (a) and by an appearance descriptor. The appearance descriptor is built from a set of feature maps pooled from the base detector (b) mapped into a lower-dimension embedding vector (c).

that compares one Anchor sample x_i^a to one Positive sample x_i^p and one Negative sample x_i^n. $f(x)\epsilon\mathbb{R}^d$ embeds a sample x in a d dimensional space and α is the margin enforced between positive and negative pairs. The loss minimizes the Euclidean distance between the Anchor and the Positive samples, while maximizing the euclidean distance between the Anchor and the Negative sample. Section 2.4.1 describes the essential step of creating the triplets training set as well as all the other implementation details.

2.3.2 Object detection linking

Our second module links detections into tracks by building a set of tubelets, i.e., sets of corresponding detections along the video, in a similar manner to [11]. This linking, summarized in Fig. 2.3, is a sequential process. We start building tubelets from the object detections between the first pair of frames and extend them as long as corresponding objects are still found in the next following frames. New tubelets can be initialized at any frame with those detections not included in existing tubelets.

To link detections between two consecutive frames, we propose a similarity function based on the following pair-wise features computed for each possible link of detections (o_t^i

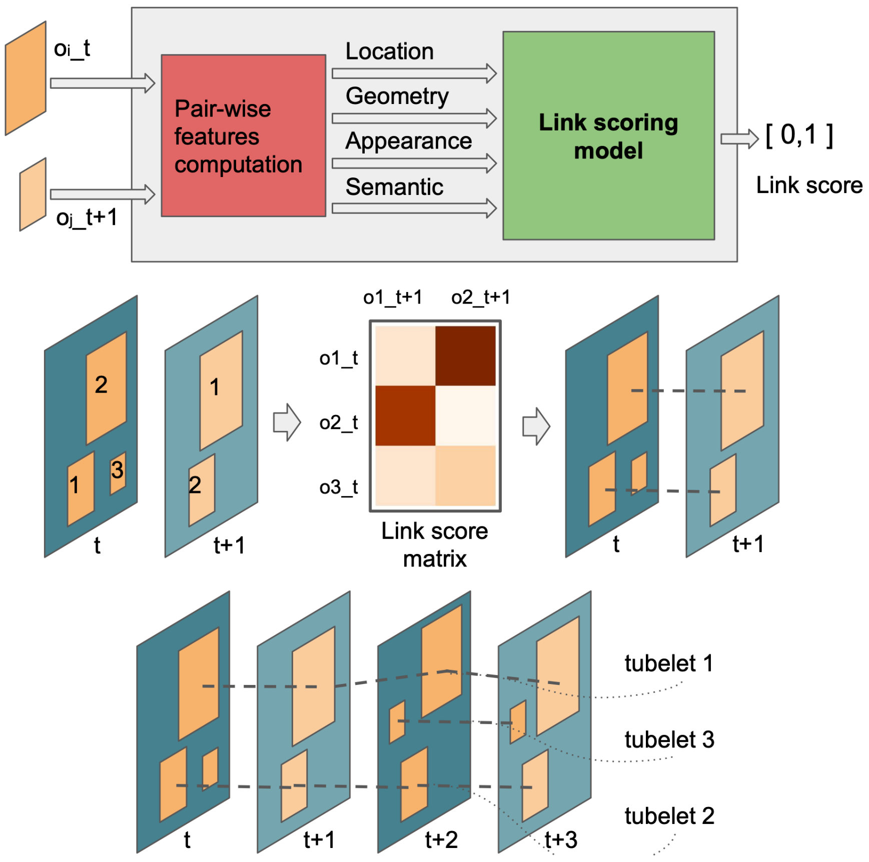


Figure 2.3: **Object detection linking**. For all possible pairs of detections from consecutive frames (t and $t+1$), we build a set of features based on their location, geometry, appearance, and semantics. These features are used to predict a linking (similarity) score. Links are established between consecutive frames and tubelets are composed of links as long as possible.

and o^j_{t+1}):

$$\begin{aligned} f_{loc} &= \{IoU, d_{centers}\}, \\ f_{geo} &= \{ratio_w, ratio_h\}, \\ f_{app} &= d_{app}, \\ f_{sem} &= f^a_{sem} \cdot f^b_{sem}, \end{aligned}$$

where IoU is the Intersection over Union of both detections, $d_{centers}$ is the relative euclidean distance between the two bounding box centers, $ratio_w$ and $ratio_h$ are the width and height ratio between the two bounding boxes, d_{app} is the euclidean distance between the appearance embeddings, and f_{sem} is the dot product of class confidences vectors (cc_t and cc_{t+1}). The actual link score (LS) between two detections is computed as follows:

$$LS(o^i_t, o^j_{t+1}) = f_{sem}\ \mathbf{X}(f_{loc}, f_{geo}, f_{app}), \tag{2.2}$$

where $\mathbf{X}$ is a logistic regression trained to distinguish whether two detections, given their pair-wise features, belong to the same object instance (high linking score) or not (low linking score). More details on how this regression is trained based on triplets information can be found in Section 2.4.1.

A score matrix is created with the link scores of every pair of detections between two frames. We use a greedy approach to match them and extend the tubelet, i.e., the highest score is picked to be a link, and the corresponding row and column are suppressed from the matrix, repeating the process until no more links can be set. Although we tried more complex methods to solve the assignment problem, such as the Hungarian method, we did not find any improvements to the algorithm described above. For each pair of frames, a new tubelet is created if an assigned pair did not belong to an already existing tubelet.

Since object detectors do not always make accurate predictions, some instance linkings are prone to generate false positives when they do not belong to any real object. When this happens, our algorithm outputs a low linking score. Linking candidates that present a score under the established threshold are filtered out.

2.3.3 Refinement: re-scoring and re-coordinating

The final step of our method uses the linkings of each tubelet to improve its object classification and location:

Re-scoring

This step simply averages all class confidence vectors from each tubelet and assigns this average to all detections within the tubelet. This process is able to correct mislabeled detections in a subset of frames or disambiguate those with low confidence.

Bounding box coordinates

Object detectors predict highly accurate bounding box coordinates on still images or on low-motion frames, but this regression tends to be less accurate on objects that present rare poses, defocus, or fast motions. We treat each coordinate of a linked object over time as a noisy time series. Note that noisy bounding box detections cannot properly fit the real object along time. We use smoothing to remove or alleviate this noise. In particular, we convolve a one-dimensional Gaussian filter along each time series. The smoothed series are then used as the set of coordinates of the object in the tubelet.

2.4 Experiments

This Section describes our post-processing evaluation on different detectors and data conditions.

2.4.1 Experimental setup

Dataset

The main dataset used is ImageNet VID [129], so far the largest densely annotated dataset for video object detection. It consists of 3862 and 555 training and validation snippets densely labeled with multiple bounding boxes belonging to 30 different object classes. These classes are a subset of the 200 classes from ImageNet DET dataset [129]. ImageNet VID data includes a wide variety of conditions, such as different object movement types, blur, defocus, or occlusions. There is an average of 1.59 objects per frame, each one with groundtruth annotations of their bounding box and track id, both in train and validation sets.

Evaluation metrics

Mean Average Precision (mAP) is computed to evaluate video object detection performance in ImageNet VID, as established in the original paper, and to compare our approach with other methods. To provide deeper insight, we also report the mAP according to three groups of motion (slow, medium, and fast) as defined in [190].

Triplet dataset used for training our models

Both the embedding model (Sec. 2.3.1) and the link scoring model (Sec. 2.3.2) are trained with the same dataset organized in *triplets*. Each triplet is composed of one Anchor (sample point), one Positive (same object instance), and one Negative (different object instance) bounding box, all of them according to ImageNet VID data groundtruth. Note that for the link scoring model each triplet provides a positive and a negative training example.

We compiled a set of 50000 and 8000 triplet samples, for training and validation respectively, built as follows. For each triplet sample, we randomly sample a track id, i.e., an object track. The Anchor is obtained from a random frame of the track. The Positive example is taken from a frame sampled from $\pm$ 25 frames away from the Anchor, i.e., within a one-second window. Note that most videos in the dataset are recorded with a frame rate between 20 and 30 frames per second. The Negative sample is simply an object with a different track id. It can be randomly obtained from either the same snippet as the Anchor or from another video. Negative examples may include objects from the same class but different instance.

Base detection model

Although our approach works with any object detector on video data, for comparison purposes we have defined a single baseline, YOLOv3 [124], as our per-frame base object detector. YOLO is a well-known and broadly used single-shot detection model that uses a Fully-Convolutional Neural Network architecture to get predictions at different scales, achieving a great time-accuracy trade-off.

This base model has been trained with data both from ImageNet DET and ImageNet VID using the same data split as FGFA [190]. We trained the model with data augmentation techniques: multi-scale input, horizontal flip, cropping, shifting, jitter, and saturation. For the inference phase, we fix the input image shape to 512x512 pixels and we replace the usual non-maximum-suppression (NMS) implementation, which works at a class detection level. Instead, we apply it at a bounding box level taking the maximum class confidence score for each detection in a frame as one of the inputs of NMS. Our NMS module finally outputs a set of bounding boxes for each frame along with a class confidence vector for each bounding box. Finally, we filter out those detections whose maximum class confidence is under a threshold of 0.005. The predictions obtained in this inference phase are the same data that we use in all the tests we perform within our ablation study and comparison with other post-processing techniques. With this configuration, we are able to obtain predictions at 23 fps with a single NVIDIA GeForce RTX 2080Ti GPU.

Appearance embedding model configuration

The feature maps used for the appearance embedding correspond to the output of the Convolutional Layers Block from the base detection model, that downsamples the input image by a factor of 16. Our RoI Pooling extracts the feature patches from these maps and scales them to a shape of 5x5x256. These values are set to minimize the loss of information on the scaling phase while allowing to predict fast appearance embeddings. The Fully Convolutional layer that learns the final embedding contains 256 neurons and outputs $L2$-normalized embeddings of 256 values.

2.4.2 Performance & analysis of other post-processing methods

Table 2.1 shows the object detection performance of our post-processing approach compared to two other well-known approaches on the ImageNetVID dataset. We evaluate the improvements of the post-processing methods with respect to the predictions obtained from a common base model (YOLOv3). We measure the mAP for all test videos and detail specific results for slow, medium, and fast-moving objects. The average processing time per frame is also shown. Each row corresponds to one of the following approaches:

- **Only base detector**. Corresponds to the execution of the base model detector, YOLOv3, with the configuration described in Sec. 2.4.1 with no post-processing.
- **Ours.** Corresponds to YOLOv3 detections post-processed with our proposed method configured with the following parameters: linking threshold of 0.7 to suppress low-scoring detection linkings, and standard deviation of the Gaussian filter in the re-coordinating module set to 0.6.
- **Seq-Bbox-Matching [11].** Corresponds to YOLOv3 detections post-processed with Seq-Bbox-Matching. Since it does not have a public implementation, we have replicated it from the paper (κ value of 12 for their tubelet linking module, as they specify).
- **Seq-NMS [52]**. Corresponds to YOLOv3 detections post-processed with Seq-NMS using the public implementation by FGFA [190], with the same parameters as in the original publication [52] but with our own confidence threshold (defined in Section 2.4.1).

The results show how post-processing is able to improve the baseline detector. Our method achieves better performance than the two state-of-the-art methods when applied to the same base detections as our approach. The computation times of our method are slightly higher than those of Seq-Bbox, but they are still in the order of milliseconds. Although our method outperforms all the groups, the difference between the methods is more noticeable for fast-moving objects, likely because both previous methods rely too much on the IoU between object detections in consecutive frames, while our approach has additional descriptors and a learned function that make the linking process more robust to changes in position. Our method improves $6\%, 8\%$, and 11% w.r.t. the baseline method, while the other methods improve around $6 - 7\%$ for all classes.

	mAP by object motion				Latency (ms)
	All	Slow	Medium	Fast	
Only base detector^{+}	68.59	76.79	66.45	45.79	44
Ours	**75.06**	**82.54**	**74.29**	**56.58**	46.58 (44 + 2.58)
Seq-NMS [52]	71.51	79.99	70.01	50.95	54.41 (44 + 10.41)
Seq-Bbox-Matching [11]	74.19	81.13	73.22	54.39	44.40 (44 + 0.40)

$^{+}$All other methods are run as post-processing of this base detector.

Table 2.1: Results of different post-processing approaches for object detection in ImageNetVID validation set. *Latency* stands for the average processing time (in ms) per frame (detection + post-processing).

To get more insight into this increased robustness to speed, we simulated a lower processing frame rate. This is important in practice when limited computational resources impede the processing at the acquisition frame rate. This more challenging scenario shows how the approaches work when objects change more drastically both their location and appearance. We ran the same object detection task on the ImageNet VID validation set, but simulating a lower processing frame rate. Since the videos have different sampling rates, we fixed the time between frames and picked the closest frame for each video. We removed tracks composed of less than 2 frames, since post-processing methods are not useful there. The number of videos dropped from 555 to 522 in the smallest data configuration (2000 ms). Note that evaluating different frames and tracks for different sampling rates can lead the mAP to not decrease as expected when the dropped data belongs to objects that are difficult to detect (this effect is more noticeable between the frame sampling periods 1000 and 2000). Since linking scores drop their value for long-time-distant detections due to their lower similarities, for this test we set a linking threshold of 0.05, more suitable to suppress spurious detections and being able to get long-term linkings. We also remove the tubelet linking module of Seq-Bbox-Matching and set the parameters of the method according to the paper recommendations for this type of experiment.

Figure 2.4 shows the mAP for all test videos, including separate plots for different object motion speeds, for varying values of frame sampling periods. We can observe similar effects to before. Our method gets comparable results to the others when working with continuous frames, but it manages to get more robust results when frames are processed more sparsely. Interestingly, our method is always able to improve over the baseline detector performance, while the other methods cannot.

2.4.3 Post-processing of video object detection methods

Table 2.2 compares the results of state-of-the-art specific video object detectors with our post-processing method applied to an image object detector (YOLOv3 [124]) and to two video object detectors (SELSA [171] and FGFA [190]). The results with YOLOv3 are the same as in the previous Table 2.1 and are included to ease comparisons. TCD [184] results can not be fully analyzed, since there is no code available up to our knowledge. We show their overall result as a reference, as reported in the original paper. SELSA and FGFA results have been obtained using their official code implementation and parameters. They

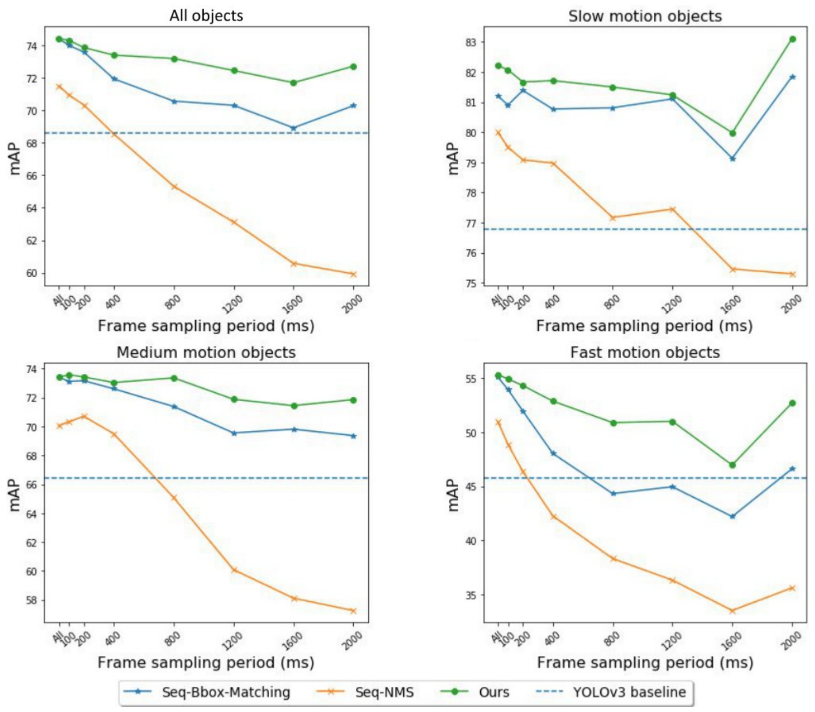


Figure 2.4: Performance of different post-processing methods evaluated with different frame sampling periods. (a) Shows mAP on all object instances. Following [190] (b), (c), and (d) show the mAP of the objects according to their motion, slow, medium, or fast. The dotted line shows the mAP baseline for each motion. All post-processing methods have been applied to our YOLOv3 baseline predictions. Seq-NMS has been calculated with the code released by [190]. Seq-Bbox-Matching has been calculated with code replicated from the paper, since the original one is not available.

use a ResNet-101 as the backbone and train with data augmentation and a mix of ImageNet VID and DET. Since extracting low-level features from a video object detector architecture is not trivial, appearance features have not been used in SELSA and FGFA tests.

Table 2.2 shows that post-processing the YOLO single image detections cannot beat the best more complex detectors that work over multiple frames. But it is interesting to note that it does achieve slightly better performance than the corresponding much more complex ResNet-based models when performing per-frame detections. However, when applied to state-of-the-art video object detectors, our approach is still able to boost the performance of SELSA and FGFA between 2 and 3%. This gain is mainly due to medium and fast object movements, where we get a 5% gain. This is not always the case for other post-processing methods that failed to improve video detectors [171].

Finally, it is worth stressing again that applying our post-processing approach is always a computationally cheap operation when compared to the detection time. Post-processing time depends on the number of detections per frame, but even with many detections it is still in the order of tens of milliseconds. Our results show that simple and efficient post-processing methods can boost the performance of many state-of-the-art detectors with minimum computational requirements and, when time constraints are important, can be a useful approach to trade-off time performance and computational resources.

	base object detector (backbone)	mAP backbone	mAP ALL	mAP slow	mAP medium	mAP fast	Latency (ms)
FGFA [190]*	R-FCN (ResNet101)	74.1	77.1	85.9	75.7	56.1	128
TCD [184]	Faster R-CNN (ResNet101)	74.6	83.5	–	–	–	–
SELSA [171]*	Faster R-CNN (ResNet101)	73.62	82.69	88.02	81.34	67.17	458
YOLOv3 + Ours	YOLOv3 (Darknet-53)	68.59	75.06	82.54	74.29	56.58	46.58 (44 + 2.58)
FGFA + Ours	R-FCN (ResNet101)	74.1	80.09	87.42	79.1	61.38	149 (128 + 21)
SELSA + Ours	Faster R-CNN (ResNet-101)	73.62	84.21	88.72	83.32	71.09	466.6 (458 + 8.6)

*Inference times are calculated running their official code on an NVIDIA GEFORCE RTX 2080Ti GPU
All models make use of both Imagenet DET and VID data for training (or pretraining).

Table 2.2: Comparison with Video Object Detectors on ImageNetVID validation set. *Latency* stands for the average processing time (in ms) per frame (detection + post-processing).

2.4.4 Qualitative results on EPIC-KITCHEN Dataset

This experiment tests our proposed post-processing pipeline in a more challenging scenario. We perform a qualitative evaluation on the EPIC-KITCHENS dataset [32]. It is an egocentric video dataset where different users perform daily activities in a kitchen. Multiple objects of different categories appear in the scenes with uncommon perspectives and movements compared to traditional video data. Groundtruth includes user action annotations, multi-language narrations, and *active* object detections (note this means not all objects are annotated in the videos).

In order to prove the generalization ability of our approach, we apply it to predictions on these egocentric videos. Predictions are obtained with a YOLOv3 model trained only with the COCO dataset [90], just still images from 80 different categories involved.

Figure 2.5 shows a scene (frames 18812, 18832, 18868 and 18908 from the video *P04_24*; note that KITCHENS videos are recorded at 60 fps) that involves a fast camera motion, where objects drastically change their location and get out of focus. The YOLO predictions (first row) suffer from many misdetections and misclassifications due to the mentioned video artifacts. The application of our approach (second row) manages to correct many of them. Partially occluded objects are detected (tracks 14314 and 14364), out of focus objects are correctly detected (track 14436) and classified (track 13020) and false positives are suppressed (observe detected knife, mouse, and toothbrush from the base predictions). By using neighboring frame information, object coordinates show a smoother evolution along the full video sequence thanks to our re-coordinating module, removing wrong shapes and flickering.

[2]Supplementary video available at: `https://github.com/AlbertoSabater/Robust-and-efficient-post-processing-for-video-object-detection`

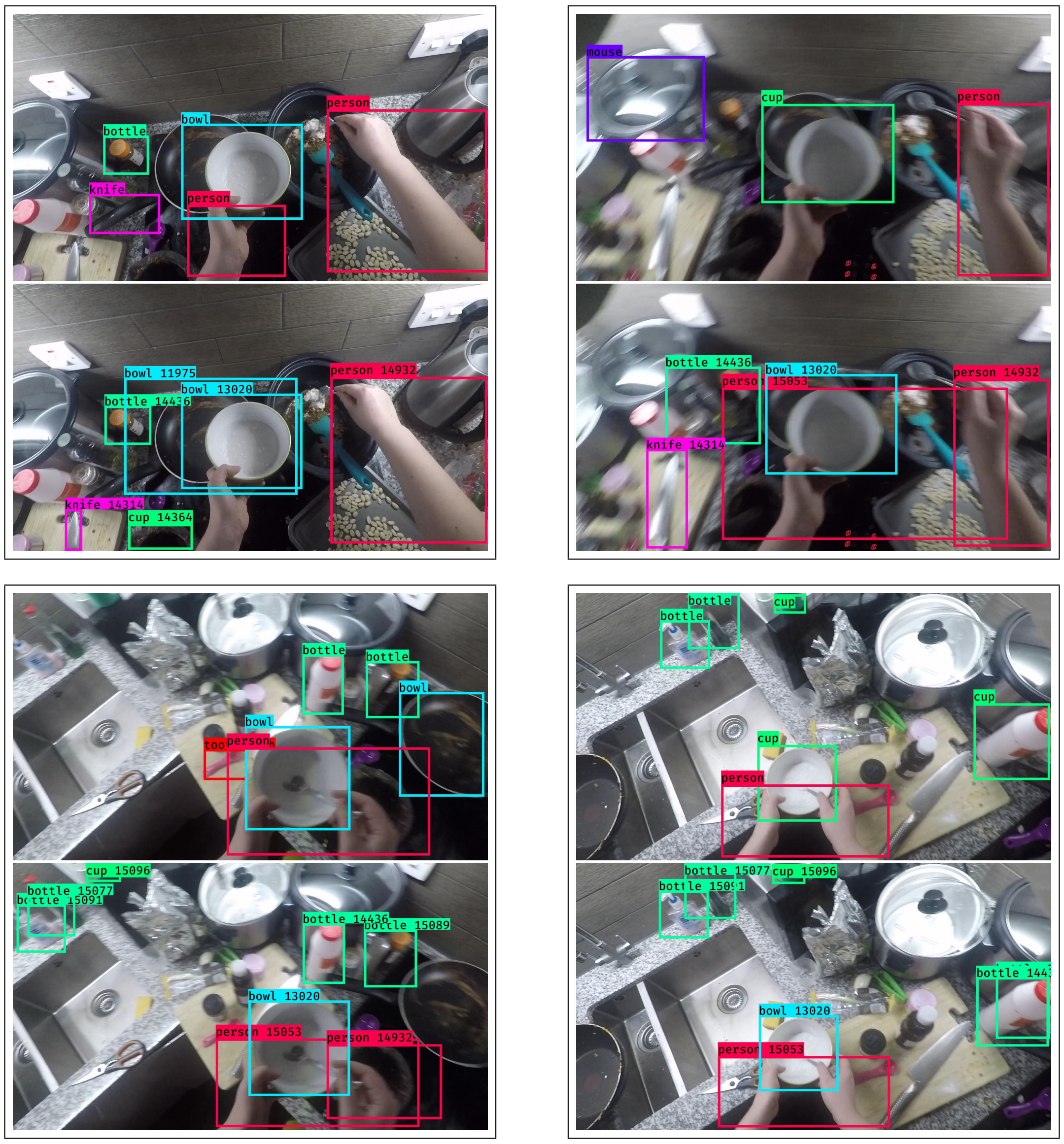


Figure 2.5: Object detection on EPIC-KITCHENS data using a YOLOv3 model (top row) and improvements obtained by applying our post-processing (bottom row). Predicted bounding boxes are shown along with their *track_id* when our post-processing is applied. Watch the supplementary video for results on the whole sequence^2. Low-scoring detections have been removed for a better visualization.

2.5 Conclusions

In this work, we present a novel post-processing method for object detection in video. By using a novel set of detection features we study the similarity between frame detections from a learning-based approach as a preliminary step to a prediction refinement. With a light computational overhead, we boost the performance of state-of-the-art video object detectors. Besides, when our approach is applied to efficient still image object detectors, we achieve comparable results to more complex models. As demonstrated, our novel post-processing solution allows to overcome significant challenges of video object detection, such as misdetections and misclassifications caused by fast-moving objects, occlusions, or de-focus. The validation presented also proves the robustness of the presented approach for videos with low frame rate, as it happens in resource-limited environments.

Chapter 3

Full-body action recognition in challenging scenarios

Recognizing actions from video data presents several challenges that hinder the effectiveness of existing approaches in real scenarios. Frequently, action recognition datasets available to learn from are limited to specific action domains and recording setups. Differently, real-world applications such as AR/VR or medical therapies might require recognizing actions that can be created on the fly and that can occur at any location of the recorded scene, regardless of the camera recording viewpoint. Since collecting and labeling this kind of heterogeneous data is complex, our goal is to learn a model invariant to these video conditions so that it can later generalize to different scenarios. The present Chapter introduces a skeleton-based solution that achieves these high generalization capabilities by designing a more robust skeleton-based representation and several inference improvements. The performance of our proposed action recognition model is demonstrated in a practical medical therapy use case where actions are defined on the fly and whose kinematic presents many high-level artifacts.[1]

3.1 Introduction

Human Action Recognition is a challenging problem with high relevance for many application fields such as human-computer interaction, AR/VR, or medical therapy analysis. Certain works use raw appearance information for the action recognition tasks [35, 182], however, most recent works use skeleton-based representations [179, 98]. These representations encode the human pose independently of appearance and surroundings and are more robust to occlusions.

Real-world scenarios often require the capability to recognize new action categories that cannot be learned, due to their creation on the fly or training data limitations. Action classifiers that handle this problem [163, 92] of learning from limited data are based on encoding motion sequences into meaningful descriptors. Then, to recognize a new target sequence, they evaluate the similarity of the target descriptor with the descriptors from one (one-shot) or few (few-shot) anchor-labeled actions. Learning discriminative action encod-

[1]Code, learned models, and supplementary video available at: `https://github.com/AlbertoSabater/Skeleton-based-One-shot-Action-Recognition`

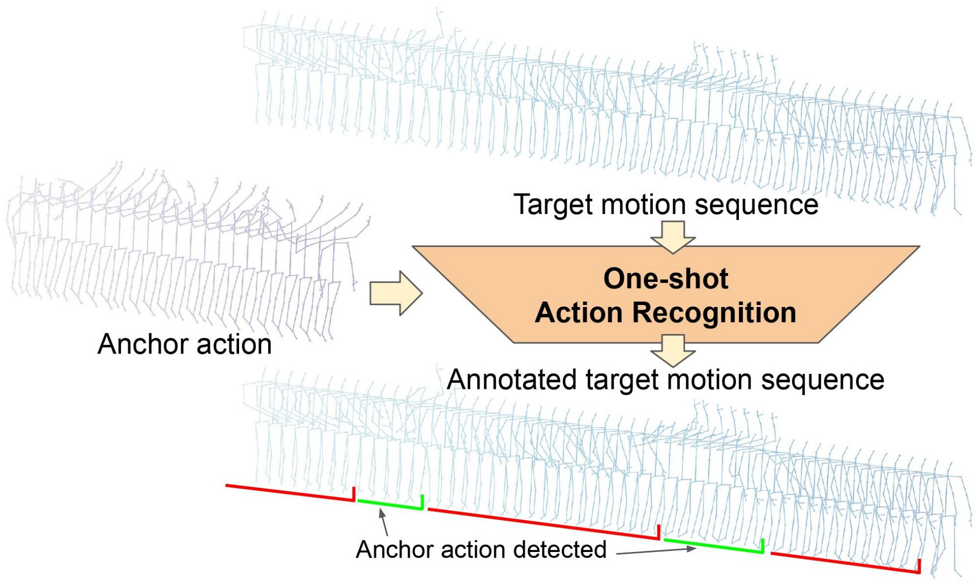


Figure 3.1: Overview of the proposed action recognition method. An anchor action is processed and compared against a target motion sequence. Output results show when the anchor action has been performed within the target motion sequence.

ings and their application in real scenarios are still open challenges. This is partially due to variable motion recording setups and unconstrained action execution rules (unsegmented action sequences, multiple consecutive executions, heterogeneous action executions, etc.).

This work tackles the problem of one-shot action recognition in unconstrained motion sequences. Our novel skeleton-based solution is summarized in Fig. 3.1. In the proposed workflow, a stream of skeleton poses is encoded in an online manner, generating a descriptor at each time step. Comparing the descriptors from a given reference anchor action with descriptors from a target video, we can detect when the anchor action has been performed within the target sequence.

The main components of our work are two-fold: 1) a motion encoder, based on geometric information, which is robust to heterogeneous movement kinematics; 2) the actual one-shot action recognition step, based on evaluating the similarity between an anchor action and target motion encodings. We propose a set of improvements to this final action recognition step designed to achieve a more accurate action recognition in the wild. These improvements include an extended anchor action representation and a dynamic threshold that discriminates challenging action sequences. Besides, the proposed action recognition approach can be easily extended to a few-shot problem, if multiple anchor actions are available.

The presented approach is validated on a generic and public one-shot action recognition benchmark, the NTU RGB+D 120 dataset [91], where it outperforms the existing baseline results. Besides, we exhaustively analyze the performance of our system on data from a real application, an automatic analysis of therapies with autistic people (based on gesture and action imitation games). The evaluation shows our proposed improvements to work in the wild, managing to overcome challenging motion artifacts that are particular for this real environment. The final outcome provides online and real-time quantitative and qualitative results, essential to evaluate patient attention and coordination.

3.2 Related Work

This Section summarizes the most relevant contributions for skeleton-based action recognition, making special emphasis on N-shot action recognition methods.

3.2.1 Skeleton representations for action recognition

Although many skeleton-based action recognition approaches use the raw Cartesian pose coordinates with no special pre-processing [117, 91], other works research skeleton data transformations to achieve view-invariant coordinates or compute new geometric features. The approach in [179] trains a Variational Autoencoder to estimate the most suitable observation view-points and transforms the skeletons to those view-points. Another approach applies view-invariant transformations to the skeleton coordinates [150]. Regarding the computation of new skeleton features, earlier work [25] computes multiple geometric features including joint distances and orientation, distances between joints, lines and planes, velocity, and acceleration. More recent approaches describe a set of geometric features based on distances between joints and lines [180] or propose to use pair-wise Euclidean joint distances and two-scale motion speeds [174].

Our approach uses a skeleton representation that combines a view-invariant transformation of the skeleton Cartesian coordinates with a set of additional geometric features.

3.2.2 Skeleton-based neural networks for action recognition

Recurrent Neural Networks have been widely applied to learn temporal dependencies for action recognition approaches. [150] uses a Variational Autoencoder with bidirectional GRU layers for unsupervised training, and work in [93] presents a recurrent attention mechanism that improves iteratively. Other approaches rely on 1D Convolutional Neural Networks (CNN), such as [174], which uses a ResNet backbone, and [179] in which a CNN is combined with LSTM networks. Another common architecture within action recognition approaches is Graph Convolutional Networks, such as the models proposed in [98] and [28], which reduce the computational complexity with flexible receptive fields.

In our method, a Temporal Convolutional Network (TCN) ([8, 157]) is chosen to encode temporal action segments into fixed-length representations. TCNs have already shown a good action recognition performance, easing the interpretability of their results [71]. TCNs use one-dimensional dilated convolutions to learn long-term dependencies from variable-length input sequences. Convolutions allow parallelizing computations allowing fast inference and performing equally or even better than RNNs in sequence modeling tasks by exhibiting longer memory [8].

3.2.3 N-shot action recognition

N-shot action recognition is still an active area of research. We find earlier methods like [37] and [73] that use HOF and HOG features for one-shot action recognition in RGB+D videos. More recently, [91] and [92] use a 2D Spatio-Temporal LSTM Network, the latter with a bidirectional pass. Other applications join visual and semantic information to perform zero-shot action recognition, such as [51], which uses hierarchical LSTMs and word2vec [105], and [65], that uses a Relation Network, a Spatial Temporal Graph Convolution Network (ST-GCN) and sent2vec. [114].

Different from these methods, we propose the simple yet effective TCN described above as a feature extractor for one-shot action recognition. Additionally, we show how a robust variation of the approach can boost the final recognition performance in challenging real-world scenarios.

3.3 Proposed Framework

Figure 3.2 summarizes our motion description approach. It first normalizes input streams of skeleton data (computed using standard techniques [183]) and generates sets of pose features that are encoded into motion descriptors by a TCN. One-shot action recognition is performed by evaluating the similarity between motion descriptors from anchor and target sequences.

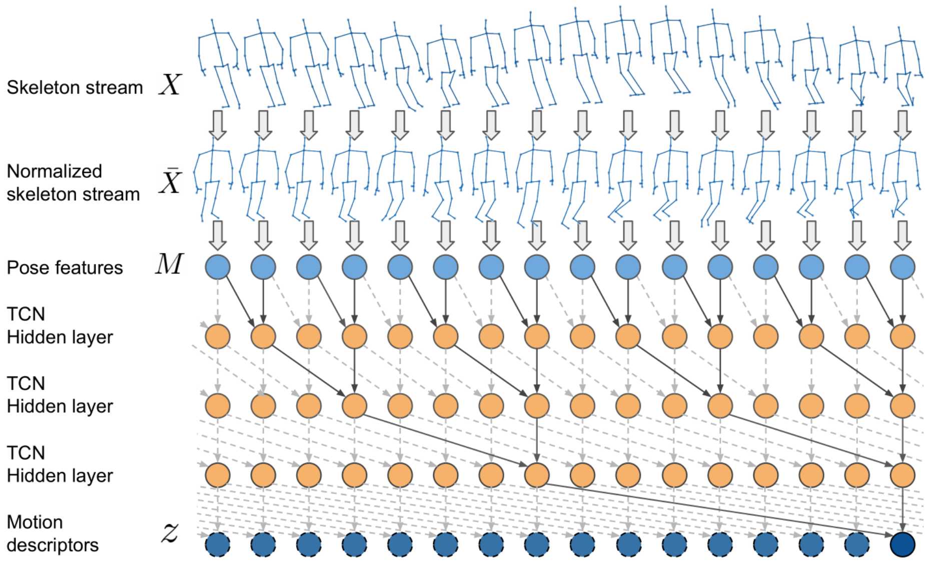


Figure 3.2: Motion descriptor generation. Input skeleton coordinates X_n are normalized $\bar{X}_n$ and pose features M are calculated. A TCN processes pose features to generate motion descriptors z.

3.3.1 Pose normalization

A human movement is defined by a set of N poses $X = \{X_1, .., X_N\}$. Each pose X_n is defined as a set of J 3D body keypoint coordinates, $X_n = \{x_1^n, ..., x_J^n\}, x_j^n \epsilon \mathbb{R}^3$, composing what we name a skeleton.

Human actions are frequently recorded in dynamic scenarios that involve different viewpoints and people moving and interacting freely. To achieve a better action recognition generalization, we normalize skeleton data by applying a per-frame coordinate transformation from the original coordinate system W to a new one H, obtaining new **view** and **location-invariant** coordinate sets. As represented in Fig. 3.3, H is set to have its origin at the middle point of the vector composed of the two hip keypoints. This hip vector is aligned to the new X axis and oriented to always have left and right hip X coordinates negative and positive values respectively. Similarly, the vector composed by the spine keypoint and the origin becomes the Y axis, leaving the Z axis to describe the depth information by being orthogonal to X-Y. The corresponding transformation T^{HW} is applied to each 3D point in a pose X_n to obtain the new set of 3D keypoint coordinates $\hat{X}_n$ as:

$$\hat{X}_n = T_n^{HW} X_n \tag{3.1}$$

Regardless of the camera configuration, action sequences can be performed by different people with heterogeneous heights. To get **scale-invariant** coordinates $\bar{X}_n$, each skeleton $\hat{X}_n$ is scaled to a predefined size. In particular, since the joints defining the torso usually present little noise, we scale each skeleton to have a fixed torso length $\bar{L}$:

$$\bar{X}_n = \hat{X}_n * \frac{\bar{L}}{L_n} \tag{3.2}$$

where L_n is the length of the corresponding torso and $\bar{L}$ is set as the average ratio between the torso length and the height (calculated from the NTU RBG+D 120 Dataset).

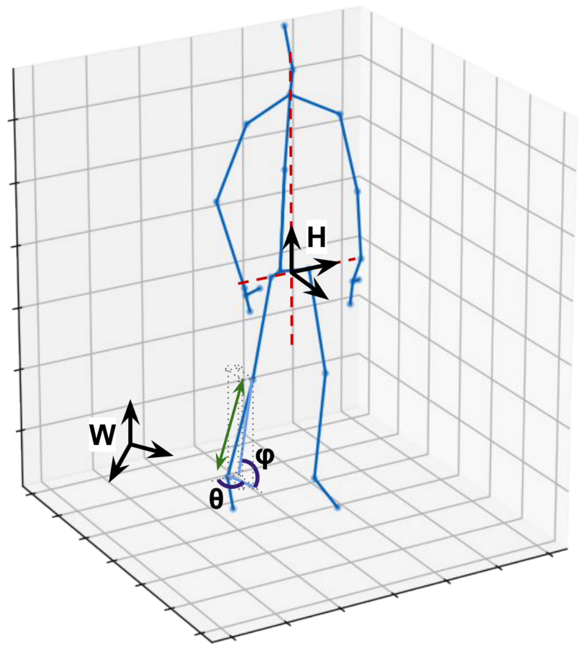
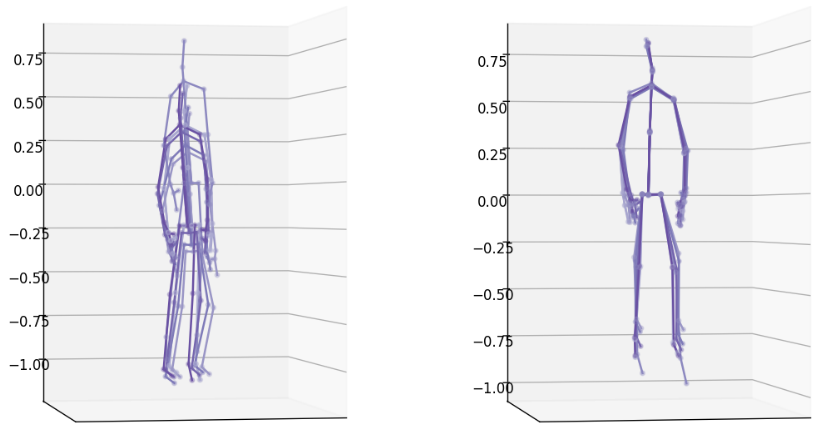


Figure 3.3: Skeleton representation. W and H refer to the original and transformed skeleton coordinates systems respectively. H axis is aligned with the vectors (dashed lines) that cross the human hip and spine. φ and θ angles refer to the elevation and azimuth calculation in a bone from the leg.

Figure 3.4 shows original and normalized skeleton coordinates after applying the two proposed normalization steps.

3.3.2 Pose Features

The presented approach includes as final pose features the normalized coordinates $\bar{X}_n$, described above, and the following additional geometric features:


(a) skeletons from *jump up* action

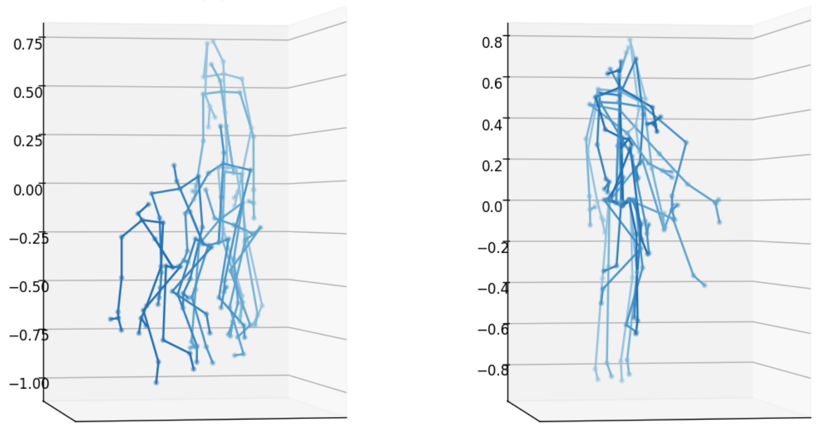

(b) skeletons from *moving heavy objects* action

Figure 3.4: Pose normalization. Left plots show original skeleton coordinates of different actions from the NTU-120 RGB-D dataset. Right plots represent the same skeletons after applying the proposed pose normalization.

- **Pair-wise keypoint distances** P_n, calculated as the Euclidean distance between each possible pair of skeleton joints, encode the pose relative to the J skeleton joints. This set of features has the size of $\binom{J}{2}$.
- ***Bone* angles** B_n from the original coordinates, calculated as the elevation φ and azimuth θ (see Fig. 3.3) of each vector composed by two connected joints. These angles encode the orientation of each bone relative to the world. This set of features has the size of $b \times 2$, being b the number of bones within the skeleton.

3.3.3 Motion descriptor generation from a TCN

In order to generate motion representations based on the temporal context and not only static pose features, we use a Temporal Convolutional Network (TCN) [8, 157]. The TCN processes, as illustrated in Fig. 3.2, streams of pose features $M = \{\bar{X}, P, B\}$, and obtains motion embeddings z (or descriptors). This motion generation works in an online fashion, creating embeddings $z_n = TCN(M_{n-w:n})$ that encode, at the time n, all the motion from the last w frames. This receptive field (memory) w is implicitly defined by the TCN hyperparameters (details in Section 3.4.1.2).

3.3.4 One-shot action recognition

We formulate the one-shot action recognition as a simple similarity evaluation between the anchor embedding z_a calculated to describe an anchor action and a stream of target embeddings z_T extracted from a full video sequence.

For the **anchor action description**, we use the embedding associated to the last frame of the anchor action, i.e., z_a, assuming it encodes all the relevant previous motion information. Then, the **evaluation distance** at time n is given by the distance between the anchor embedding and the target embedding $z_T(n)$ computed at time n:

$$d_1(n) = D(z_a, z_T(n)) \tag{3.3}$$

For the **embedding distance computation** D we have explored several options. The cosine distance (cos) and the Jensen-Shannon divergence (JS) are the two best performing ones:

$$D_{cos}(z_1, z_2) = 1 - \frac{z_1 \cdot z_2}{\|z_1\|\|z_2\|}, \tag{3.4}$$

$$D_{JS}(z_1, z_2) = \sqrt{\frac{KL(z_1 \,||\, \bar{z}_{1,2}) + KL(z_2 \,||\, \bar{z}_{1,2})}{2}} \tag{3.5}$$

where z_1 and z_2 are two motion embeddings, KL is the *Kullback-Leibler divergence* and $\bar{z}_{1,2}$ is the pointwise mean of z_1 and z_2. Both functions are bounded between 0 and 1, being this last value the lowest similarity between two movement descriptors.

The **final action recognition** is performed by thresholding the calculated distance. If $d_1(n)$ is below the acceptance threshold α, we consider that the anchor action has been detected by frame n. This threshold value is set by evaluating the precision/recall curve over an evaluation dataset, as detailed in Section 3.4.3.1.

3.3.5 Improving action recognition in the wild

Real action recognition applications (e.g., real medical therapies) involve artifacts that hinder the motion description and recognition. These issues are intensified in the one-shot recognition setup, where the available labeled data is limited. In the following, we describe different improvements for the action recognition described previously, to get better performance in the wild.

Extended anchor action representation. Anchor actions are not only scarce but frequently also hard to consistently segment in a video sequence. Therefore, using just the last embedding generated for them can lead to noisy action descriptions. In order to get a better anchor action representation, we use a set of descriptors $z_A = \{z_1, ..., z_m\}$ composed by the ones generated at their last m frames. The distance between a target embedding $z_T(n)$ and the anchor action is then set as the minimum distance to each element of the set of anchor embeddings (z_A):

$$d_m(n) = \min_{\forall z_a \in z_A} D(z_a, z_T(n)) \tag{3.6}$$

Few-shot recognition. In case more than one anchor sequence is available, the set of anchor embeddings z_A can be easily augmented with the embeddings generated for the different anchor sequences.

Dynamic threshold. Most challenging scenarios can include actions consisting of maintaining a static pose or performing subtle movements. The descriptors generated from this type of action are very similar to the descriptors from target idle poses (e.g., when no specific action is being performed). If the information on idle positions is available, we propose to use it to set a dynamic threshold that can better discriminate idle poses from actual actions. The new threshold value is set to be the minimum between the original threshold α and the 10^{th} percentile $P_{10\%}$ of all the distances computed between a given idle target sub-sequence (identified within the target sequence) and the anchor action representation:

$$\alpha = \min(\alpha, \underset{n=a...b}{P_{10\%}} \{d(n)\}), \tag{3.7}$$

where a and b are the initial and final time steps of the idle interval, and d is computed as in Eq. (3.4) or Eq. (3.6) depending on the anchor representation used.

3.4 Experiments

This Section details the experimental validation of the proposed action recognition approach, including implementation and training details.

3.4.1 Experimental setup

3.4.1.1 Datasets

The proposed method is designed for one-shot online action recognition in real scenarios. In particular, our motivation is to automate the analysis of medical therapies. Since the available data in this setting is scarce, real therapy data is only used for evaluation, while a large public action dataset (NTU RGB+D 120) is used to learn the action encoding model as well as to validate the motion representation framework in a public benchmark.

The **NTU RGB+D 120** dataset [91] (see Fig. 3.5a) is so far the largest dataset available for skeleton-based action classification. It contains 114,480 action samples belonging to 120 different categories with different complexities. Action sequences are recorded at 30 fps with different users, setups, and points of view.

The **therapy dataset**[2] (see Fig. 3.5b) has been acquired in real medical settings and consists of 57 different imitation games where the patient has to imitate the therapist [136, 135] actions and gestures. Each imitation game is composed of at least one anchor action and one patient imitation, but frequently more anchors (up to 3) or imitations appear in the games. Due to the nature of this data, patient motion is characterized by strong artifacts such as highly variable length imitations, uncommon resting poses and poor quality imitations. Due to limited computational resources, sessions are recorded with a variable low frame rate of 12 fps on average.

3.4.1.2 Implementation and training details

Due to the nature of the therapy dataset, we have specified the memory length w of our TCN to be 32 frames long by using a convolutional kernel size of 4, stacks of 2 residual

[2]`https://zenodo.org/record/4700564#.ZCq75nZByUk`

(a) NTU RGB+D 120 dataset frames.

(b) Therapy dataset frames.

Figure 3.5: Sample frames from the training (a) dataset (*kicking something*, *handshaking* and *shoot at the basket actions*) and evaluation (b) datasets (*big*, *give* and *high gestures*).

blocks, and dilations of 1, 2, and 4 for the layers within each convolutional block. We also use skip connections and 256 filters in each convolutional layer, which finally generates motion descriptors of size 256.

This TCN is pretrained for a classification task (categorical cross-entropy loss) in the NTU RGB+D 120 dataset by adding an output classification layer to the TCN backbone and applying the following random data augmentation to the original coordinates:

- **Movement speed variation**. Joint coordinates are randomly scaled by interpolation along the time dimension to simulate varying movement speeds.
- **Skip frames**. Since the training dataset is recorded at more than twice the frame rate of the evaluation dataset, for each training sample we just use one out of either two or three frames, discarding the rest.
- **Horizontal flip**. Coordinates that correspond to the left or right part of the body are randomly flipped. Vertical and depth dimensions remain as originally.
- **Random cropping**. Actions longer than the receptive field are randomly cropped to fit the TCN memory length. Shorter ones are pre-padded with zeros.

Finally, after calculating and stacking all the pose features M the input training data has a size of 32×423.

After training, the output classification layer is discarded to extract the 256-dimensional motion descriptors straight from the TCN backbone.

3.4.2 Validation of the system on a generic one-shot action recognition benchmark

Table 3.1 shows the classification accuracy of our framework for one-shot action recognition on the NTU-120 one-shot 3D action recognition problem, where we outperform the results of previous action recognition models [91]. For this benchmark, we use the same implementation described in Section 3.4.1.2 and the same training procedure, but with the train/test splits specified in the original paper [91]. Additionally, we also set the frame skipping to 2 both for training and evaluation and we suppress the random horizontal flipping during training. Descriptors are evaluated with the cosine distance, but other distance functions such as the Jensen-Shannon divergence report identical or comparable performance. Since this problem is about classifying action segments and not identifying anchor actions in the wild, classification is performed by assigning, for each evaluation sample, the class with the lowest similarity distance among the set of anchor action segments. Note that, unlike the other action recognition models that work in an offline fashion, our model works in an online and real-time fashion and only uses half of the available data (alternate frames).

Method	**Accuracy**
ST-LSTM [92] + Average Pooling	42.9%
ST-LSTM [92] + Fully Connected	42.1%
ST-LSTM [92] + Attention	41.0%
APSR[91]	45.3%
Ours (one-shot, m=1)	**46.5** %

Table 3.1: NTU RGB+D 120 Dataset One-shot evaluation

3.4.3 Validation and discussion on real therapies data

This experiment evaluates the performance of our action recognition approach on imitation games from the therapy dataset. Here, the actions from the therapist are taken as anchors to later detect their imitation at each time step of the patient motion stream.

3.4.3.1 Quantitative results

Metrics. Action recognition is evaluated with common metrics, i.e., precision, recall, and F1. However, since each target motion descriptor $z_T(n)$ does not refer only to a single time-step n but to the motion from the w previous frames, we have defined the following terms as follows:

- **True Positive (TP).** An action is correctly detected when the groundtruth action is being executed or has been recently executed within the TCN receptive field w. A groundtruth action referenced by many detections only counts as one TP.

- **False Positive (FP)**. An action is incorrectly detected when no groundtruth action has been recently executed. Consecutive detections inside the TCN receptive field w only count as one FP.

- **False Negative (FN)**. A groundtruth action is missed if it has not been referenced by any action detection. Each miss-detected action counts as a single FN.

Precision-Recall trade off. The threshold α selected to be used over the distance scores to perform action recognition is the one that optimizes the trade-off between precision and recall defined by the F1 metric. In our experiments, we calculate different thresholds to optimize the results achieved when using one descriptor per anchor ($m = 1$) and with extended anchor representations ($m = 3$). As seen in the precision-recall curves from Fig. 3.6, the best trade-off is achieved at the cost of lowering the precision of the framework, especially when working with single anchor descriptors.

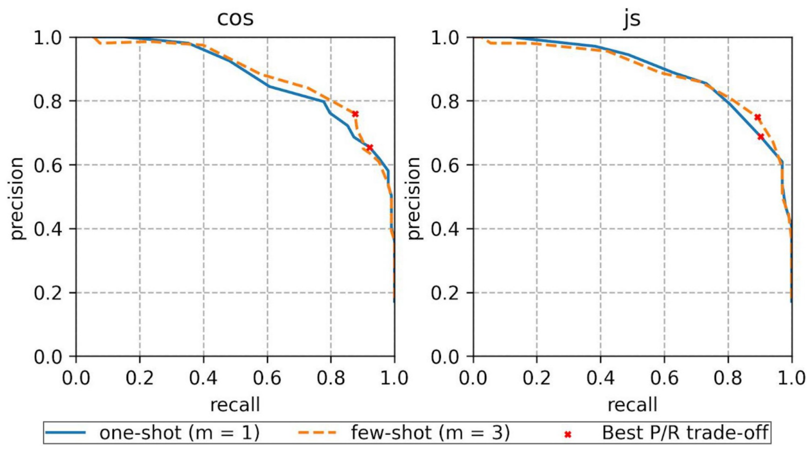


Figure 3.6: Precision/recall curves for the two evaluation distances used: cosine distance (cos) and Jensen-Shannon divergence (js). Solid lines refer to regular one-shot evaluations ($m = 1$) and dashed lines refer to few-shot recognition with extended anchor representations ($m = 3$). Red crosses refer to the optimal F1 values.

Recognition Modality	**m**	**Dyn. Thr.**	**Cosine distance**				**Jensen-Shannon divergence**			
			α	**P**	**R**	**F1**	α	**P**	**R**	**F1**
one-shot	1		0.48	0.655	0.922	0.697	0.50	0.689	0.902	0.714
one-shot	3		0.40	0.773	0.837	0.724	0.48	0.765	0.853	0.728
one-shot	3	✓	≤ 0.40	0.803	0.837	**0.751**	≤ 0.48	0.779	0.853	**0.739**
few-shot*	1		0.48	0.784	0.827	0.731	0.50	0.785	0.846	0.744
few-shot*	3		0.40	0.760	0.876	0.753	0.48	0.749	0.892	0.755
few-shot*	3	✓	≤ 0.40	0.790	0.876	**0.781**	≤ 0.48	0.763	0.892	**0.765**

$*$ few-shot uses 1 to 3 anchor sequences depending on its availability in each test (see Section 3.4.1.1)

Table 3.2: Comparison of the studied variations of our action recognition approach on the evaluation therapy dataset. *Dyn. Thr.* stands for *Dynamic Threshold*. *P* and *R* stands for *Precision* and *Recall*.

Analysis of variations of our approach. The effect of the different variations proposed to make our action recognition more robust is summarized in Table 3.2. First three rows show the performance of the one-shot version, i.e., using as anchor the last action performed by the therapist in each imitation game. The use of extended anchor representations ($m = 3$) allows us to have a more restrictive threshold α, which reduces false positives, increasing the detection precision. The increase in precision is also achieved by making the threshold dynamic. New threshold values can be set differently in each imitation game by evaluating the anchor representation and the target motion up to the end of the anchor execution (period in which the patient is not performing any specific action). The dynamic threshold avoids false positives related to "static" anchor actions, at no recall cost. Finally, as expected, the performance improves when more anchor actions (up to 3 in the experiments) are available to run a few-shot recognition. Regarding the embedding similarity computation, both the cosine distance and the JS divergence report similar performance when using extended anchor representations. However, the JS divergence performs better in simpler setups and the cosine distance excels when using a dynamic threshold.

The results can be further scrutinized in Table 3.3, which compares the recognition performance by action category. Extended anchor representations ($m = 3$) and few-shot recognition significantly overcome the limitations of the simple one-shot recognition for challenging classes (last 8 rows of the table refer to actions with softer and subtler arm movements or actions that consist in staying on specific positions). This is further improved when using a dynamic threshold (DT).

Class name	**One-shot (m=1)**	**Few-shot (m=3)**	**Few-shot+DT (m=3)**
big	1.00	1.00 (+0.00)	1.00 (+0.00)
high	1.00	1.00 (+0.00)	1.00 (+0.00)
happy	0.96	0.96 (+0.00)	0.96 (+0.00)
waving	0.89	0.95 (+0.06)	0.95 (+0.06)
pointing	0.89	0.89 (+0.00)	0.89 (+0.00)
giving	0.76	0.76 (+0.00)	0.76 (+0.00)
small	0.68	0.89 (+0.21)	0.96 (+0.28)
coming	0.65	0.69 (+0.04)	0.69 (+0.04)
waiting	0.63	0.73 (+0.10)	0.73 (+0.10)
hungry	0.47	0.35 (-0.12)	0.42 (-0.05)
where	0.47	0.61 (+0.14)	0.61 (+0.14)
down	0.45	0.54 (+0.08)	0.54 (+0.08)
me	0.44	0.67 (+0.22)	0.67 (+0.22)
angry	0.31	0.30 (-0.01)	0.50 (+0.19)

Table 3.3: Per-class F1 evaluation comparison of different variations of our action recognition approach (using the cosine distance).

Influence of different pose feature sets. Table 3.4 reports the influence of different pose features proposed in Section 3.3.2. Using only normalized coordinates achieves lower performance than using just the original body coordinates. Normalized coordinates gain in invariance, but they lose certain discriminative power. The proposed geometric feature set achieves good performance on its own. However, the best performance comes

Coordinates	Normalized Coordinates	Geometric Features	F1
✓			0.711
	✓		0.689
		✓	0.757
✓		✓	0.695
	✓	✓	**0.781**

Table 3.4: F1 evaluation of different pose features running our top-performing action recognition (few-shot, cosine distance, extended anchor representations, and dynamic threshold).

from the combination of normalized coordinates and geometric features, which suggests that the geometric features work better along with the invariance provided by normalized coordinates.

3.4.3.2 Qualitative results

Figure 3.7 shows the timeline of an imitation game sampled from the therapy dataset. This exercise involves the action of raising and moving the arms, which is performed once (anchor) by the therapist and repeated twice (target) by the patient. Detection results from the timeline represent both the location and quality of both action repetitions. The latter is estimated according to the calculated similarity between anchor and target motion. The first action imitation is performed quickly (worse quality) and the second one is slower and more detailed (better quality). For a better understanding, the whole execution of the experiment is shown in the supplementary video[3].

3.4.3.3 Time performance

Temporal Convolutional Networks are frequently lightweight models, making them suitable for low-resource environments like in the therapies described in Section 3.4.1.1. With the settings described in Section 3.4.1.2, the action recognition (except the skeleton extraction from RGB-D data) takes 0.08 ms per time-step using the GPU and 0.1 ms just using the CPU. Time performance has been calculated with the therapy dataset skeleton sequences. Note that due to the TCN parallel computing, the longer the motion sequences are, the faster per-frame processing we can get. Latency has been calculated with an NVIDIA GeForce RTX 2080 Ti (GPU) and an Intel Core i7-9700K (CPU).

3.5 Conclusions

This work presents a novel skeleton-based framework for one-shot action recognition in the wild. Our method generates motion descriptors robust to different motion artifacts and variable kinematics. We achieve accurate action recognition by combining the proposed set of pose features and an efficient architecture based on a TCN that encodes these pose features. Besides, we demonstrate the effectiveness of several simple steps included in our method to boost the action recognition performance in challenging real-world scenarios,

[3]Supplementary video available at: `https://github.com/AlbertoSabater/Skeleton-based-One-shot-Action-Recognition`

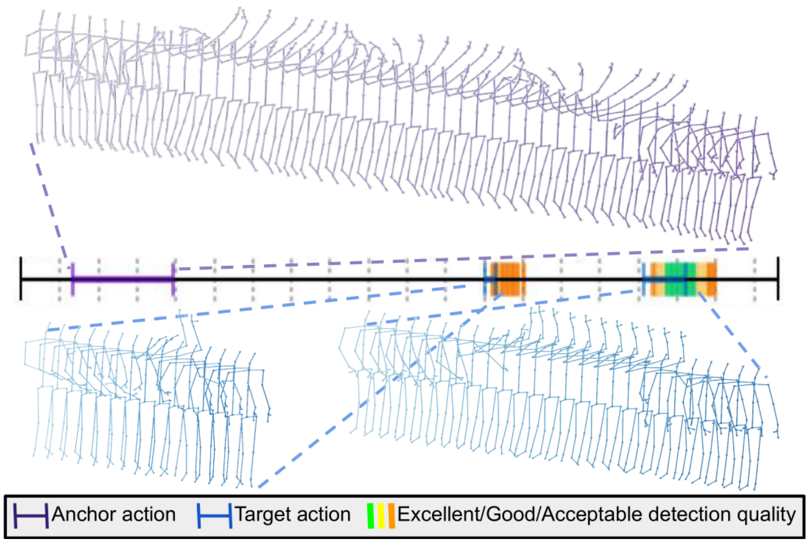


Figure 3.7: Action recognition on one imitation game from the therapy dataset. Central horizontal line represents the execution timeline, with one anchor action (top/purple) performed by the therapist, followed by two imitations (bottom/blue) performed by the patient. Dashed line segments indicate the groundtruth action time segmentations. Vertical small solid segments, ranging from green to orange, refer to time steps where the action has been detected, and the color gives an estimation of the repetition quality.

i.e., with limited reference data and noisy repetitions of the actions to be recognized. Our base one-shot recognition approach is evaluated on the public NTU RGB+D 120 dataset, outperforming previous action recognition methods, by using only half of the available data (alternate frames) for training. We also demonstrate the suitability of our framework to analyze videos from real therapies with autistic people. These recordings are characterized by having extreme motion artifacts given by the difficulty of the patients to pay attention and their limited coordination abilities. Evaluation results provide quantitative and qualitative measures of the actions recognized, essential to evaluate and monitor aspects like the time the patient takes to imitate an action and the number of tries required to do it with good quality. Moreover, our approach can run online, being able to provide immediate feedback to the therapist and making the sessions more dynamic.

Chapter 4

Hand action recognition invariant to domain and viewpoint

As discussed in the previous Chapter, real scenarios require action recognition models to be robust to work with heterogeneous actions. Our prior work introduced a new way of representing skeleton data that abstracts the camera recording viewpoint to help achieve these generalization capabilities for full-body action recognition. However, this representation format is not applicable to hand-only skeletons since the recording perspective is required to define certain hand actions. For instance, with no external references, the gesture of *pointing to the right* recorded with a first-person view can be misidentified as *pointing to the left* if the recording camera is in front of the person. Additionally, generalizing to unseen hand actions presents additional challenges. When working with hand-only motion data, there are subtle changes in actions from a specific domain, but significant variations across different ones (e.g., virtual reality vs. life-logging applications). This makes models trained on specific domains unable to properly generalize to different ones. This Chapter addresses these issues by considering the recording perspective in the skeleton data representation and in the learning pipeline, and includes an improved model for the final motion description. The proposed hand action recognition framework not only gets high accuracy for the regular action recognition on the training motion setup, but we also probe its generalization capabilities by evaluating it on the recognition of unseen action domains and recording perspectives.[1]

4.1 Introduction

Human action recognition is a well-studied problem with many applications such as human-robot interaction, surveillance, and monitoring [76, 154]. Deep models combined with skeleton-based representations [107, 173], which efficiently encode human pose and motion, independently of appearance, surroundings, and occlusions, have become a standard in robust human action recognition [117, 179].

Hand action recognition is a specific case of human action recognition. It is highly relevant due to the importance of hand movements in teamwork, assistive technologies,

[1]Code, learned models, supplementary video, and data splits available at: `https://github.com/AlbertoSabater/Domain-and-View-point-Agnostic-Hand-Action-Recognition`

communication, or in virtual reality applications [1, 10]. Hand recognition methods combine, as in the full body case, skeleton representations and deep models [174, 181]. These methods have shown good results typically focusing on the classification of actions from a specific domain. However, previous works have not studied robust representations that can generalize across different domains and viewpoints, which are key when working with limited amounts of labeled data.

Hand actions expose some specific challenges to learning such robust representations. On one hand, there is a high variability across actions from different domains, e.g., user interface control vs. life-logging applications. Differently from full-body skeletons, different hand action domains often imply drastic viewpoint changes, e.g., egocentric vs. third-person view. On the other hand, fine-grained details are essential. Different action categories are often quite similar and vary only subtly (e.g., pointing to different directions, sliding gestures, etc.). Moreover, hand skeleton joints present a lower movement range than other full-body joints, increasing the correlation between skeleton joint motions and similar actions.

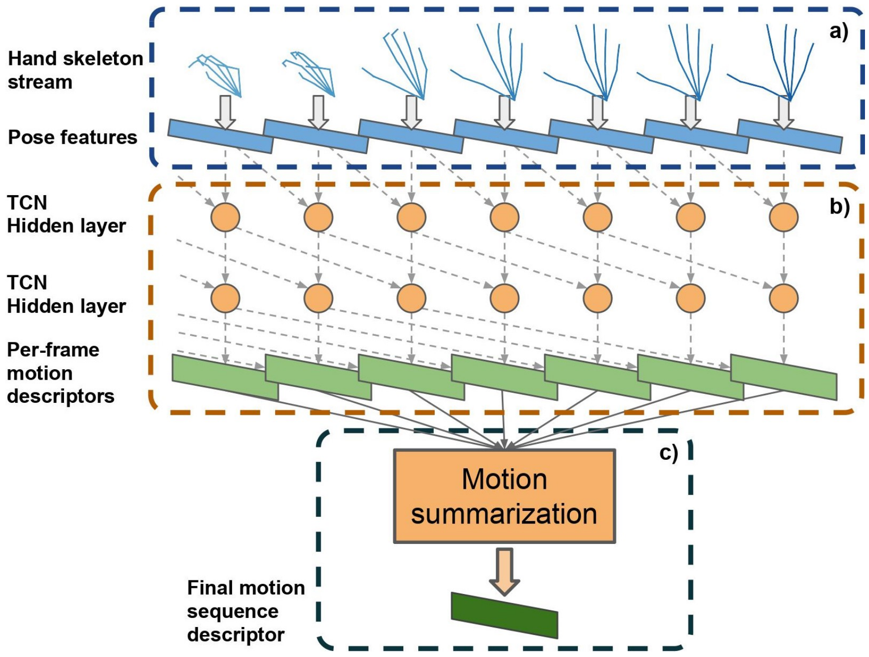


Figure 4.1: Motion representation model. **(a) Hand pose modeling**: pose features are extracted from the input hand skeleton stream. **(b) Motion representation**: a Temporal Convolutional Network (TCN) generates per-frame motion descriptors from the pose features. **(c) Motion summarization**: per-frame motion descriptors are weighted and summarized to generate the final motion sequence descriptor.

The main contribution of this work is a novel motion representation model, summarized in Fig. 4.1, designed to be robust to different application domains and viewpoints. It computes representations (motion descriptors) from labeled hand skeletons (motion sequences), that are later used for the final motion sequence classification. The main components of our motion representation model are: 1) a set of pose features adapted to hand motion; 2) a Temporal Convolutional Network (TCN) encoding the stream of hand pose features into per-frame descriptors; and 3) a summarization module that learns the relevance of each per-frame descriptor to describe the input motion sequence. The learned motion descriptors can be directly used to recognize the action categories (labels) they were trained for (intra-domain) with a simple Linear Classifier. More interestingly, they can also be used to build N-shot classifiers to recognize new unseen action categories recorded from radically different points of view (cross-domain) with a K-Nearest Neighbor Classifier.

Our experiments use the front view SHREC-17 dataset [33], the egocentric F-PHAB dataset [43] and the third-person MSRA [151] dataset, which includes actions and gestures related to computer interaction, life-logging and sign language domains respectively. Our intra-domain classification results show that our framework gets better or similar performance than current state-of-the-art intra-domain classifiers in well-known benchmarks. More importantly, our cross-domain classification approach obtains comparable accuracy to intra-domain methods by being trained just with the SHREC-17 dataset, and then evaluated on the F-PHAB and MSRA datasets. This demonstrates that our motion representation model generalizes well for different action domains and camera viewpoints. Besides, our approach shows a low latency, which allows its use for online and real-time applications.

4.2 Related Work

This Section summarizes relevant works on the core topics of this work: pose modeling, skeleton-based action recognition models, and generalization to unseen action categories.

4.2.1 Pose modeling for action recognition

Action recognition was first tackled by directly analyzing RGB videos [39] or depth maps [113]. Current approaches have settled the standard of extracting the intermediate representation of skeleton poses [181, 174]. This representation has shown great performance since it encodes human poses regardless of their appearance and surrounding and presents strong robustness to occlusions.

Certain works directly use the raw coordinates of skeleton joints (position of the joints in the Euclidean space) as input for full-body action recognition [117, 91] and for hand action recognition [57, 102, 84]. In order to achieve standardized and generic skeleton pose descriptions, several full-body action recognition approaches propose different strategies, such as learning the most suitable viewpoint for each action [179] or transforming all coordinates to a common coordinate system [134, 150]. However, this kind of transformation cannot be directly applied to hand action recognition, where orientation plays a key role.

In order to get more informative pose representations than the raw joint coordinates, many approaches propose computing additional geometric (pose) features. Chen et al. [25] use static features (distance and angles of pairs of joint coordinates) and temporal features (velocity and acceleration of joint coordinates). Zhang et al. [180] calculate distances between joints and planes, and Yang et al. [174] use joint distances and their motion speeds at different scales.

Our approach proposes a simplification of the skeleton representation reducing coordinate redundancy by using just a set of key joints. Then, simplified skeleton coordinates are standardized by applying scale and location invariant transformations. Specific geometric features are calculated to encode relevant translation information (lost in the standardization) and orientation-aware information.

4.2.2 Action recognition models

As in many other fields, deep learning has become state-of-the-art in action recognition. Particularly relevant for this work, Recurrent Neural Networks (RNN) have been widely used to model temporal dependencies in hand action recognition. Ma et al. [102] use an LSTM-based Memory Augmented Neural Network to model dynamic hand gestures. Chen et al. [27] use an LSTM Network to combine skeleton coordinates, global motions, and finger motion features. Li et al. [84] combine a bidirectional Independently Recurrent Neural Network with a self-attention-based graph convolutional network.

Other works make use of Convolutional Networks. Liu et al. [94] recognize posture and action by using 3D convolutions. Yang et al. [174] use 1D convolutions to process and fuse different hand motion features. Hou [57] proposes to focus on the most informative hand gesture features by using a ResNet-like 1D convolutional network with attention.

Our method uses a Temporal Convolutional Network (TCN) [8, 157] that implements 1D dilated convolutions to learn long-term temporal dependencies from variable-length input sequences, achieving comparable or better results than RNNs [8]. TCNs have demonstrated good performance on full-body action recognition, both with unsupervised learning [150] and supervised learning [134, 71].

4.2.3 Generalization to unseen action categories

Learning a model able to classify unseen categories is a challenging task. It is commonly addressed by encoding every new data sample into a descriptor and using a K-Nearest Neighbors classifier (KNN) to evaluate and assign labels according to the similarity between a few new category reference samples and the target samples [166].

Several works [91, 134] address this problem for action recognition by extracting intermediate feature maps from a supervised action recognition model. Koneripalli et al. [74] train an autoencoder to learn these descriptors in an unsupervised fashion. Ma et al. [102] learn these descriptors directly in a semi-supervised manner by training an encoder with metric-learning techniques. Other works use word2vec [51] and sent2vec [65] approaches for descriptor learning.

Previous works [91, 51, 65] are aimed to recognize unseen full-body action categories where no drastic camera view-points are found. Up to our knowledge, generalization to unseen hand viewpoints and domains is still to be studied. The present work uses metric learning and specific data augmentation to learn meaningful hand sequence descriptors. Our framework performs accurate action recognition of sequences from unseen categories and recording viewpoints.

4.3 Hand action recognition framework

The core of the proposed framework is the motion representation model summarized in Fig. 4.1. First, our approach calculates specific pose features for each skeleton (in our case already pre-computed and available in public datasets, see Section 4.4.1.1). These features are fed to a Temporal Convolutional Network to generate a set of motion descriptors. Additionally, a motion summarization module combines them, according to their relevance, into the final motion representation. In the following, we describe these steps, as well as how to train our motion representation model, both for intra-domain and cross-domain classification.

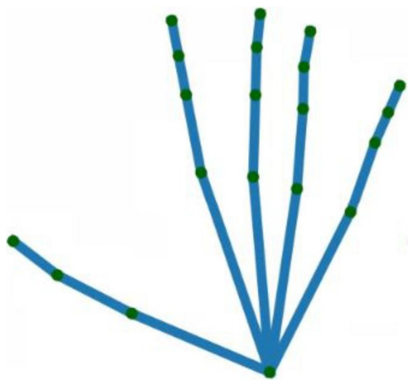
4.3.1 Hand pose modeling

Human hand motion sequences are defined by sets of T hand skeleton poses $X = \{X_1, ..., X_T\}$, extracted from video frames. Each hand skeleton X_t is composed of a set of J joint coordinates, $X_t = \{x_1, ..., x_J\}, x_j \epsilon \mathbb{R}^3$ (i.e., the position of the joints in Euclidean space), which are logically connected by a set of B bones (see Fig. 4.2).

4.3.1.1 Skeleton standardization

Since motion information has high variability across different action domains, we propose several steps to standardize the skeleton representation to help generalization of the motion representation model.

First, hand joints belonging to the same bones (fingers) are highly coupled and can be represented with a smaller number of degrees of freedom. Based on this assumption, we propose to use just a subset of **7 joints to define a hand pose** (see Fig. 4.2), corresponding to the wrist, the top of the palm, and the tips of the 5 fingers; which we connect with a total of **6 hand bones**, one for the palm and one more for each one of the fingers. This simpler skeleton representation makes the learning process easier and less prone to overfitting.


(a) 20-joint hand skeleton

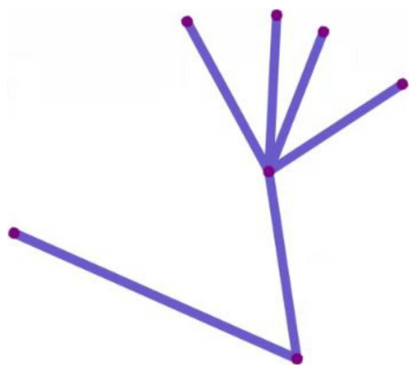

(b) 7-joint hand skeleton

Figure 4.2: Hand skeleton simplification. a) refers to a detailed hand skeleton of 20 joints (dots) connected by 19 bones (lines). b) refers to our proposed hand skeleton simplification of 7 joints (dots) and 5 bones (lines).

Secondly, since actions can be performed by different people with heterogeneous hand sizes and recorded at different scales, we standardize each skeleton pose X_t to achieve **scale-invariant** skeleton representations $\hat{X}_t$ by applying, to all the hand coordinates, the transformation that makes the palm of size equal to 1:

$$\hat{X}_n = \frac{X_n}{|P|}, \tag{4.1}$$

where $|P|$ is the euclidean distance between the wrist and the top of the palm (both joints included in original and simplified 7-joint formats).

Finally, since actions must be recognized regardless of the position where they are executed, we compute **location-invariant** coordinates (relative coordinates) by translating the top of the palm to the origin of the reference coordinate system. Note that these relative hand coordinates describe properly the intra-relation of the hand joints, but they are now missing the information related to the hand motion direction.

4.3.1.2 Hand pose description

Different from full-body motion sequences (e.g., walking) where their movement direction can be inferred from the relative coordinates of its bones (e.g., legs), hands can be translated through any direction without any change of their relative coordinates. Since the translation information is essential in certain actions (e.g., pointing to specific directions), we generate extra **translation and orientation-aware** features from the original hand skeletons:

- **Difference of coordinates**, defined as the difference of each joint coordinate with itself in the previous time-step. These features describe the translation direction and speed of each coordinate for each of the 3 axes:

$$d_{coord}(t, j) = x_{j,t} - x_{j,t-1}, \forall j \epsilon J, t \epsilon T \tag{4.2}$$

- **Difference of bone angles**, defined as the difference of the elevation φ and azimuth θ of a bone $b \epsilon B$ with itself in the previous time-step. These features describe the rotation direction (with respect to the world coordinates) and rotation speed of each bone:

$$d_{\varphi}(t, b_{\varphi}) = b_{\varphi,t} - b_{\varphi,t-1}, \forall b \epsilon B, t \epsilon T \tag{4.3}$$

$$d_{\theta}(t, b_{\theta}) = b_{\theta,t} - b_{\theta,t-1}, \forall b \epsilon B, t \epsilon T \tag{4.4}$$

Our final hand representation is a feature vector of size 54 (Fig. 4.1.a) (7×3 relative hand coordinates, 7×3 coordinate difference features, and 6×2 bone angle differences).

4.3.2 Motion representation model

The core of our action recognition framework is a model that encodes the skeleton features from each frame, described in the previous Section, into single motion descriptors with a Temporal Convolutional Network (TCN) [8, 157]. The TCN processes sequences of skeleton features, generating a descriptor at each time-step per-frame descriptors) that represents the motion performed up to that frame, i.e., with no information from the future (see Fig. 4.1.b).

For a given motion sequence, the last descriptor generated by the TCN is frequently the one used to represent the action [134] since it encodes all the information up to that point. However, training motion sequences are not frequently segmented in time with high precision. In these cases, sequence endings contain frames that are not informative for the action they represent. Consequently, the last descriptor can introduce some noise that hinders the training.

To alleviate this issue, we learn the relevance of the temporal patterns of the actions. More precisely, we add a **motion summarization module** after the TCN (see Fig. 4.1.c), which combines all the per-frame descriptors generated for the input hand motion, up the

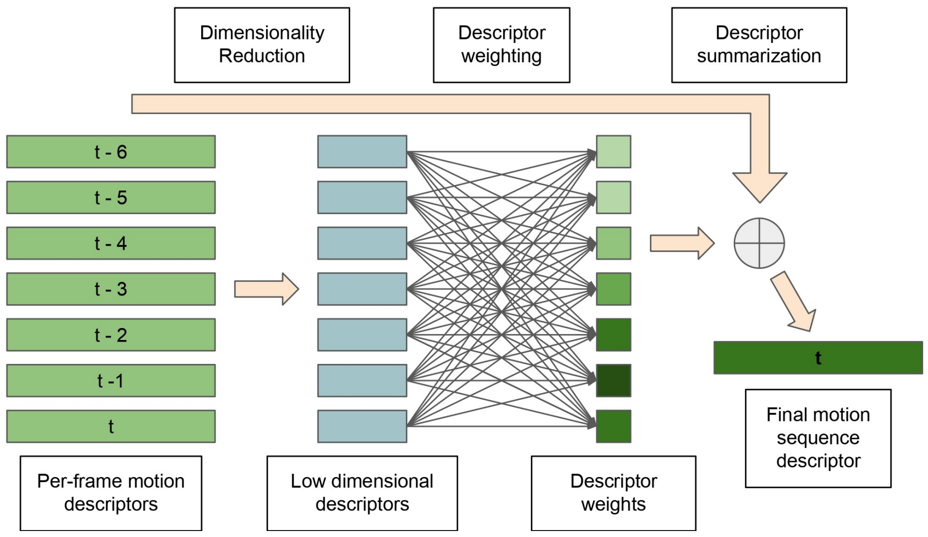


Figure 4.3: Motion summarization module. Per-frame motion descriptors are simplified with a 1D Convolutional layer. They are grouped with a single perceptron layer to calculate their summarization weights. The motion sequence is summarized into a final motion descriptor by performing a weighted average over the initial per-frame motion descriptors.

TCN memory length, by performing a weighted average over them (details in Fig. 4.3). These weights represent how important each descriptor is for the final motion representation. They are learned with a simple Neural Network trained end-to-end along with the TCN. This network consists of a single 1D Convolutional layer with kernel 1 that reduces the dimensionality of the per-frame descriptor, and a single Fully Connected layer with a sigmoid activation layer, that takes as input all the simplified descriptors and outputs a vector of categorical probabilities (i.e., descriptor weights). These final weights are L1 normalized before performing the final descriptor summarization.

This summarization module efficiently describes hand motion sequences and helps the TCN to focus just on meaningful data during training. However, there are real use cases where actions, at test time, present a longer length than our motion representation module can handle. In these cases, although the summarization module has been trained along with the TCN, it is better to discard it and classify individually all the per-frame descriptors generated by the TCN, which still contain meaningful motion representations.

So far, we have shown how to encode a motion sequence X into a robust simple descriptor $z = f(X)$, where the function f represents our motion representation module (Fig. 4.1). In the next two sections, we describe how to optimize these motion representations to perform intra-domain classification and cross-domain classification.

4.3.3 Intra-domain classification

Intra-domain hand action classification aims to recognize the same action categories (labels) seen during the learning phase, with no drastic variation in the camera viewpoint. For this classification, intra-domain class probabilities $P = g(z)$ are predicted by a linear classifier g trained end-to-end along with our motion representation model f (represented in Fig. 4.1). Intra-domain classification is learned by the optimization of the categorical cross-entropy loss:

$$CCE = -\Sigma_{c=1}^{C} y_{i,c} \log \left(p_{i,c}\right), \tag{4.5}$$

which evaluates the predicted probabilities $p_{i,c}$ that belongs to a class $c \epsilon C$, given their true label y_i.

Each training iteration includes a mini-batch composed of motion sequences sampled uniformly for each action category (2 different samples per category in our experiments). To ensure the generalization to different motion artifacts, which can be hard to achieve with small datasets, each motion sequence within the mini-batch is included three times with different data augmentations. This data augmentation is applied to the per-frame skeletons X_t, before the feature computation from Section 4.3.1, as follows:

- Movement speed variation. Joint coordinates are randomly re-sampled by interpolation over the temporal dimension. This simulates different motion speeds, and thus, different sequence lengths.
- Frame skipping. Since contiguous video frames contain similar joint information, we only use one out of every three frames, reducing the data redundancy and making the learning process easier. Motion sequences are then initialized randomly between the three first frames.
- Random cropping. When the sampled motion sequence is longer than a defined maximum length (i.e., TCN memory length), it is randomly cropped.
- Random noise. Gaussian noise is added to the skeleton coordinates to simulate inaccurate joint estimations.
- Random rotation noise. The whole motion sequence is rotated randomly over the 3D axes. This rotation is limited to low angles, to simulate just subtle variations in the recording viewpoint.

4.3.4 Cross-domain classification

Cross-domain hand action classification aims to recognize motion sequences whose action category and recording camera viewpoint were not present in the training data. To obtain view-point agnostic motion representations, our motion representation model f is trained, via contrastive learning, to project motion descriptors in a space where descriptors belonging to the same action category (label) must be close to each other (similar descriptors), and far away from other category descriptors (dissimilar descriptors). This is achieved by optimizing the *normalized temperature-scaled cross-entropy loss* (**NT-Xent**) [26]:

$$l_{i,j} = -\log \frac{\exp\left(\operatorname{sim}\left(z_i, z_j\right)/\tau\right)}{\sum_{k=1}^{2N} 1_{[k \neq i]} \exp\left(\operatorname{sim}\left(z_i, z_k\right)/\tau\right)}, \tag{4.6}$$

which is computed at each training iteration for each pair of actions i and j that belong to the same action category. NT-Xent maximizes the cosine similarity *sim* of both motion descriptors z_i and z_j and minimizes their similarity to the descriptors related to different action categories k. τ is a temperature parameter.

The training of our motion representation model is performed with the same batch construction and data augmentation techniques described in Section 4.3.3. Additionally, we add an extra data augmentation step that rotates randomly all the motion sequences of the mini-batch over the three axes. This batch augmentation simulates arbitrary camera recording perspectives, which is crucial to boost the performance achieved with the NT-Xent loss in different domains and camera viewpoints.

Once this generic motion representation model has been trained on a given source domain, we use an N-shot approach [166] and generate motion descriptors for a small set of N reference motion sequences (motion reference set) from a different target domain, with no specific training on the latter. To perform action classification in this new domain, we use a simple K-Nearest Neighbors classifier (KNN) to assign a label to new sequences depending on their descriptor distance to the descriptors from the motion reference set. To improve the performance of the KNN, we extend our motion reference set by applying the same data augmentation strategies described in Section 4.3.3, and we compute descriptors for all the new augmented sequences.

4.4 Experiments

This Section details the datasets used in the evaluation and our implementation details. Then, we expose the main framework design choices and evaluate its performance for cross-domain and intra-domain action recognition. Finally, we evaluate the time performance of the presented approach.

4.4.1 Experimental setup

4.4.1.1 Datasets

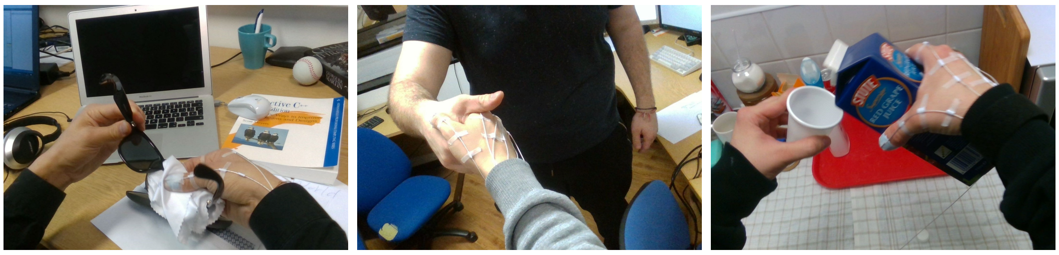
The presented approach has been validated on three different datasets (see frame samples in Fig. 4.4), with different application domains and camera viewpoints.

SHREC-17 [33] contains motion sequences (22-joint hand skeletons) related to human-machine interaction domains recorded from a **frontal third-person view**. The data is categorized with two levels of granularity, presenting 14 and 28 action categories respectively. The dataset contains 1960 motion sequences for training and 840 sequences for validation. Actions are performed by 28 different users.

F-PHAB [43] contains motion sequences (21-joint hand skeletons) recorded from an **egocentric view** related to kitchen, office, and social scenarios, which involve the interaction with different objects. Actions have been performed by 6 different users and labeled with 45 action categories. The dataset consists of 1175 motion sequences which are split into training and validation as stated by the authors [43]: **1:3, 1:1, 3:1** splits the motion sequences on different training:validation ratios (e.g., in the 1:3 split, 33% of the data is used for training and the remaining 66% is used for validation); **cross-person** 6-fold leave-one-out cross-validation, one fold for each user motion sequence. Only the original cross-subject and 1:1 splits are available, for the other two data partitions we create three random data folds to perform 3-fold cross-validation.

MSRA [151] contains motion sequences (17-joint hand skeletons) of 17 different American Sign Language gestures performed by 9 different users. Each gesture sequence has a length of 500 frames recorded from a **third-person view**. For the classification of this data, we use the motion samples from the two first subjects as reference, leaving the remaining seven as the target samples, as suggested in [94].

(a) SHREC-17 depth sample frames


(b) F-PHAB RGB sample frames

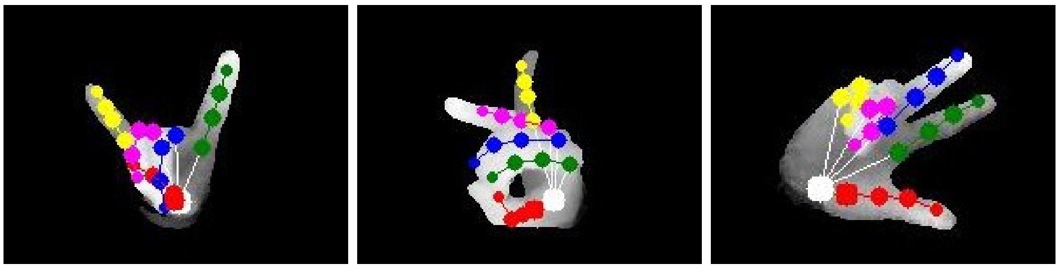

(c) MSRA depth sample frames with skeleton joints

Figure 4.4: Sample frames from the different evaluated data domains. (a) SHREC-17 dataset. Examples of actions *grab*, *expand*, and *rotation clockwise*. (b) F-PHAB dataset. Examples of actions *clean glasses*, *handshaking*, and *pour juice*. (c) MSRA dataset. Examples of the signs *IP*, *RP*, and *three*.

4.4.1.2 Implementation and training details

Hand skeleton Since each dataset used provides a different skeleton joints format, we use the 20 joints that SHREC-17 and F-PHAB have in common (see Fig. 4.2), and our proposed 7-joint skeletons representation, described in Section 4.3.1, suitable for the three datasets considered.

Motion representation architecture our motion representation model backbone is a TCN with two stacks of residual blocks with dilations of 1, 2, and 4 for the layers within each block, and convolutional filters of size 4, making a memory length of 32 frames long. Since the feature pre-processing filters out 2 out of 3 consecutive frames, this memory length covers 96 real frames. Our backbone uses 256 filters in each convolutional layer, generating motion sequence descriptors of size 256. The summarization module reduces their dimensionality to 64 with a single 1D convolutional layer and then a single perceptron layer of size 32 generates the final descriptor weights. When the sequence summarization module is not used, the descriptor generated by the TCN at the last motion time-step is used for the action representation (*Last TCN descriptor*). τ from Eq. 4.6 is set as 0.07.

KNN classifier Our KNN classifier weights pairs of target-reference descriptors according to the inverse of their distance. We validate the use of different numbers of neighbors, i.e., 1, 3, 5, 7, 9, 11, and we report the results of the neighbor that optimizes the final classification accuracy. Additionally, the reference augmentation step increases the reference descriptors set randomly up to 40 times.

4.4.2 Framework design evaluation

This subsection analyzes and validates the main components of our framework using the **cross-domain approach** of Section 4.3.4 since this setup is more demanding in terms of generalization capabilities. We train our base motion representation model on the front view SHREC-17 dataset. Then, we evaluate its accuracy on the egocentric F-PHAB validation splits (described in Section 4.4.1.1).

To analyze the effect of different design choices, we start representing the motion sequences with the last descriptor generated by the TCN at the last time steps (no use of the motion summarization module).

First, we show the benefits of using our proposed hand skeleton simplification. Table 4.1 shows in each column the accuracy obtained in each of the F-PHAB validation splits. Our proposed simplified 7-joint skeleton format reduces the coordinate redundancy and facilitates the generalization to other domains by reducing the overfitting on the source one. From now on, we set the 7-joint skeleton format as default.

Skeleton size	**1:3**	**1:1**	**3:1**	**cross-person**
20 joints	63.8	69.9	69.8	51.4
7 joints	66.3	71.0	73.8	53.5

Table 4.1: **Influence of the number of skeleton joints** in the hand representation. Motion representation model trained on SHREC. Action recognition accuracy validated on F-PHAB.

Table 4.2 shows the influence of using different classes to discriminate motion sequences while training the motion representation model. Higher class granularity (28 action categories) manages to improve the cross-domain performance by learning more informative motion descriptors. From now on we set this class granularity as default for training.

Results from Table 4.3 show how our summarization module, from now on set as the default motion representation method, improves the classification accuracy with respect to the last descriptor of our TCN backbone. The summarization module suppresses noisy and

SHREC categories	**1:3**	**1:1**	**3:1**	**cross-person**
14	58.3	65.4	65.9	48.9
28	66.3	71.0	73.8	53.5

Table 4.2: **Influence of the training categories.** Motion representation model trained on SHREC with 7-joint skeletons. Action recognition accuracy evaluated on F-PHAB.

non-informative per-frame descriptors, achieving a more informative motion representation. Interestingly, our motion summarization module learns good motion representations even when not many reference actions are available (splits 1:3). Another interesting finding is that augmenting the motion reference set helps to increase the accuracy in all the data splits by a noticeable margin.

Action descriptor	**1:3**	**1:1**	**3:1**	**cross-person**
Last TCN descriptor	66.3	71.0	73.8	53.5
Summarization	70.6	75.5	77.7	58.4
Summarization*	76.2	79.7	82.0	62.7

* includes an augmented motion reference set

Table 4.3: **Influence of the motion sequence summarization technique.** Motion representation model trained on SHREC (7-joint skeletons and 28 labels). Action recognition accuracy evaluated on F-PHAB. *Last TCN descriptor*: descriptor generated by the TCN at the last time-step. *Summarization*: descriptor generated by our summarization module.

However, we still find an accuracy drop when generalizing to actions of users not present in the motion reference set (cross-person splits). This is due to the high inter-subject action variability of the F-PHAB dataset, and because no data from this dataset has been used to train our representation model.

Figure 4.5 shows the weights learned by our summarization module on the F-PHAB validation split (1:1). This plot illustrates the intuitive idea that later per-frame descriptors are more informative than earlier ones for final motion sequence representation. However, computed weights do not exhibit a continuous growth over time, probably because contiguous time descriptors contain similar information. Although final descriptors may encode information about the whole action, they may also encode motion not related to the action itself but with idle poses for example. Therefore, they are not always the most relevant for the final action representation.

4.4.3 Cross-domain action classification

This experiment evaluates the cross-domain generalization of our framework by classifying motion sequences from action categories and camera viewpoints not seen in the training data. For this experiment, we train our motion representation model as defined in Section 4.3.4 only on the front view SHREC-17 dataset (28 labels), and we evaluate it on the egocentric F-PHAB dataset. Results from our framework correspond to the processing of 7-joint skeletons and the use of our proposed motion summarization module.

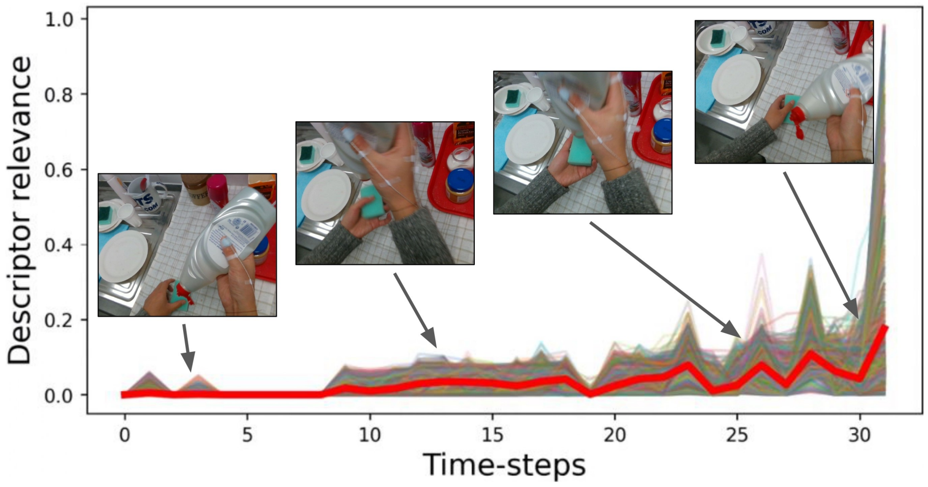


Figure 4.5: Relevance weights generated by the summarization module for each action sample from the F-PHAB validation split (1:1). Red line on top shows the average of all generated weights. Frames correspond to the action of *pour liquid soap*.

Table 4.4 shows the accuracy of the best-performing methods on the F-PHAB dataset, trained as an intra-domain problem (upper block), and the results of our cross-domain approach (bottom block). The latter includes the evaluation of DD-Net [174], one of the best-performing methods on the SHREC-17 classification benchmark. We used the available public code to train it with the SHREC-17 dataset (20-joint skeletons) as the authors state, extracting F-PHAB descriptors from its backbone and classifying them with our N-shot approach. Results from its evaluation show a lack of domain adaptation. Our method clearly outperforms the rest in this scenario.

Model	**1:3**	**1:1**	**3:1**	**cross-person**
RGB [39]	–	75.3	–	–
Depth [113]	–	70.61	–	–
LSTM [189]	58.75	78.73	**84.82**	62.06
DD-Net [174]	75.09	81.56	88.26	**71.8**
Gram Matrix [181]	–	85.39	–	–
Two-stream NN [84]	–	**90.26**	–	–
DD-Net [174]	59.6	63.7	67.5	51.2
Ours	70.6	75.5	77.7	58.4
Ours*	**76.2**	79.7	82.0	62.7

* includes an augmented motion reference set

Table 4.4: F-PHAB accuracy comparison. Upper block: results for methods trained on the F-PHAB dataset (*intra-domain* classification). Bottom block: methods trained on SHREC-17 dataset (*cross-domain* classification).

The results show that our approach clearly outperforms the RGB [39] and depth-based [113] models trained on the target domain. It is noticeable that we also get better or comparable results than a regular LSTM network [189] trained on the target dataset, especially when not many reference actions are available (1:3 split) or when not all the subjects are present in the reference split (cross-person splits). Although our cross-domain performance is behind the best intra-domain classification model [181], we show later in Section 4.4.5 that we outperform them when training in the same domain. Remember that no specific training with the F-PHAB data splits has been performed in our evaluations.

4.4.4 Cross-domain classification of long video sequences

In this experiment, we use the MSRA dataset, with hand motion sequences much longer than the memory of our representation model. This helps to illustrate two characteristics of our method. First, the motion summarization module (Fig. 4.1.c) not only helps to summarize the input motion but also to enforce the TCN to generate informative per-frame descriptors (Fig. 4.1.b). Second, per-frame descriptors can also be used to describe the input motion at each time step and perform online and real-time recognition (see Section 4.4.6)

This experiment uses the same model trained in Section 4.4.3. We evaluate its cross-domain performance in the MSRA dataset. Since sequences are too long for our summarization, we perform the KNN classification of all the motion descriptors generated by the TCN at each time-step (4.1.b), denoted as *online action classification.* We report the average of class probabilities of the frames within a video sequence for comparison with previous works, denoted as *video classification.* For computational reasons, we randomly select just 8000 reference descriptors for the KNN evaluation.

Table 4.5 shows that, even though MSRA motion sequences do not correspond to the kind of motion seen in the training data, our approach achieves a high online per-frame classification. Moreover, a simple average of the predicted frame probabilities results in a 97.1% accuracy, comparable to current state-of-the-art results specifically trained on the MSRA dataset. In this case, reference motion data augmentation does not provide an edge, probably because MSRA motion sequences already contain enough hand pose variations.

Model	**Online classification**	**Video classification**
3D PostureNet [94]	–	98.56
Ours	85.8	97.1
Ours*	86.7	97.1

* includes an augmented motion reference set

Table 4.5: MSRA accuracy comparison. Batched vs. online predictions. Our results correspond to our motion representation model trained on SHREC-17 data (cross-domain).

4.4.5 Intra-domain classification and reference actions study

This experiment evaluates our method for intra-domain classification using the linear classifier from Section 4.3.3.

4.4.5.1 SHREC-17 evaluation

Table 4.6 shows the classification accuracy of our framework trained and evaluated on the SHREC-17 dataset. Results show that, even though our method was designed for cross-domain classification, it gets comparable results to the state-of-the-art when trained with the target dataset. Note that we are using just 7 out of the 22 original skeleton joints, which helps generalization to other datasets but it might lose domain-specific information.

Model	**SHREC 14**	**SHREC 28**
DD-Net [174]	94.6	91.9
Two-stream NN [84]	96.31	94.05
Ours	93.57	91.43

Table 4.6: Intra-domain classification on SHREC-17.

4.4.5.2 F-PHAB evaluation

Table 4.7 shows the classification accuracy of our framework trained and evaluated on each one of the F-PHAB data splits. DD-net results are obtained by training on the F-PHAB dataset with the original code and following the original paper [174]. Results show how we manage to outperform the current state-of-the-art in all the splits. Interestingly, our method excels even when less training data is available (1:3). This generalization is visible even on the high inter-subject variability (cross-person) [43].

Model	**1:3**	**1:1**	**3:1**	**cross-person**
DD-Net [174]	75.09	81.56	88.26	71.8
Two-stream NN [84]	–	90.26	–	–
Ours	**92.90**	**95.93**	**96.76**	**88.70**

Table 4.7: Intra-domain classification on F-PHAB.

4.4.6 Time performance

The presented work is a lightweight solution, able to perform online and real-time hand action recognition (like in Section 4.4.4). Our base motion representation model (4.1.b) gets per-frame descriptors in 0.8 ms per time-step in GPU (NVIDIA GeForce GTX 1070) and 1 ms in CPU (Intel Core i7-6700). The motion summarization module (Fig. 4.1.c) and the linear classifier (intra-domain) from Section 4.3.3 can be used at a negligible cost. The KNN classifier (cross-domain) from Section 4.3.4 has a cost $O(k * log(n))$ that depends on the number of neighbors k and the size n of the motion reference set. For instance, the KNN classification on the 1:1 data split from F-PHAB (575 motion reference sequences), just takes 0.2 ms per motion descriptor when using 5 neighbors and 0.5 ms when augmenting the motion reference set 40 times.

4.5 Conclusions

This Chapter introduces a hand action recognition solution, specifically designed to be robust to different action domains and camera perspectives, and able to perform in online and real-time domains. Our solution consists of a framework that, given a set of skeleton motion sequences, extracts sets of pose features adapted to heterogeneous motion kinematics.

Then, our motion representation model uses a Temporal Convolutional Network to generate per-frame motion descriptors, and a simple motion summarization module weights them, according to their relevance, generating the final motion representation. The proposed motion representation model is trained and validated in the following two problem setups. For intra-domain classification, we achieve better or similar results than state-of-the-art methods in well-known benchmarks. More importantly, in cross-domain classification, our approach is able to generalize to unseen target action domains and camera viewpoints, achieving comparable results to the state-of-the-art methods trained on the target data domains.

Chapter 5

Scene Recognition with event cameras

Previous Chapters were based on the analysis of RGB video data or skeleton coordinates extracted from it. This Chapter studies the use of event cameras and develops methods to process the data that these sensors record. Different from traditional cameras, event sensors are able to efficiently capture just the sparse changes in the scene, ignoring the redundant static information and, similar to skeleton coordinates, abstracting the appearance of the scene for better generalization. Moreover, event cameras present high robustness to challenging lighting conditions and high temporal resolution. The starting hypothesis for our work is that prior event-based work does not sufficiently benefit from the event camera efficiency properties. This leads to deep learning models that, although achieving good accuracy, present a level of efficiency not suitable for low-resource environments. In the following, we analyze the potential of the event cameras and design a framework that efficiently benefits from the event data properties, while achieving high performance. Our framework is tested for event stream and dense per-pixel estimation, and includes an extension to perform multi-modal learning, demonstrated with an application that learns jointly from event data and grayscale images.[1]

5.1 Introduction

Event cameras are bio-inspired sensors that register, with minimal power consumption, changes in intensity at each pixel of the sensor array. Contrary to traditional cameras, they work in a sparse and asynchronous manner, with an increased high dynamic range and high temporal resolution (in the order of microseconds). These characteristics have pushed the research of many event-based perception tasks such as action recognition [15, 61], body [18, 128] and gaze tracking [6], depth estimation, [45, 167] or odometry [72, 126], interesting for many applications that involve low-resource environments and challenging motion and lighting conditions, such as AR/VR or autonomous driving.

Processing information from event-based cameras is still an open research problem. Top-performing approaches transform event streams into frame-like representations, ignor-

[1]Code, trained models, and supplementary video available at: `https://github.com/AlbertoSabater/EventTransformer` and `https://github.com/AlbertoSabater/EventTransformerPlus`

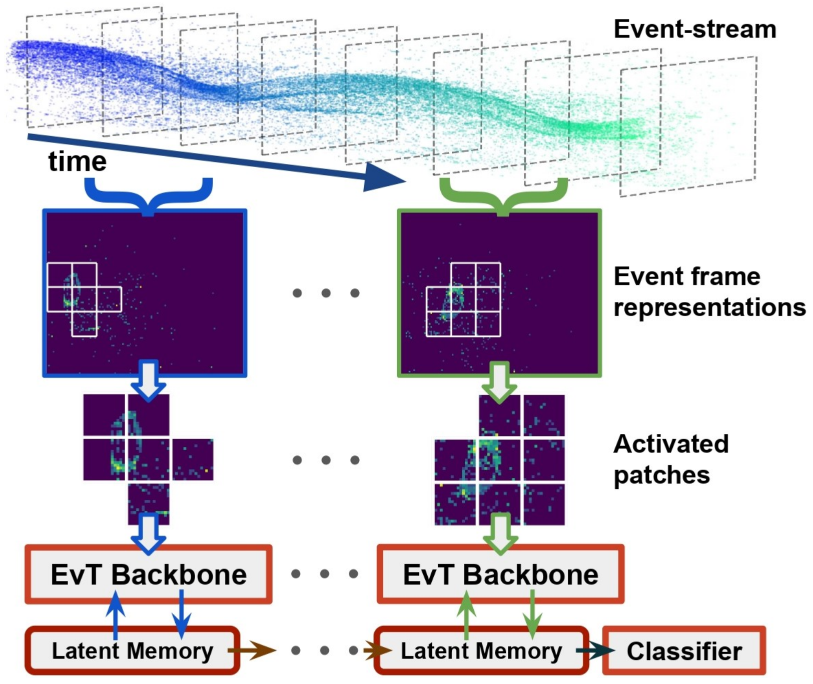


Figure 5.1: **Event Transformer and Event Transformer$^+$ overview**. Areas (*activated patches*) from event streams with sufficient event information are extracted from event frame representations and processed by the EvT orEvT$^+$ backbone to update a set of *latent memory vectors*. The latest version of this memory is used for a final event stream classification.

ing their inherent sparsity, and using standard heavy processing algorithms such as Convolutional Neural Networks [5, 61, 9, 19] and/or Recurrent Layers [61, 19]. Other methods that better exploit this sparsity, such as PointNet-like Neural Networks [165], Graph Neural Networks [15, 34] or Spike Neural Networks [67, 145], are more efficient but do not reach the same accuracy. As a result, there is a need for methods that put to good use all the potential of event-based cameras for efficiency and low energy consumption while maintaining high performance.

This Chapter introduces two models, *Event Transformer* (*EvT*) and its extension *Event Transformer*$^+$ (*EvT*$^+$), designed to tackle the event data sparsity and be highly efficient while obtaining top-accuracy results. Both methods are based on two new proposed ideas (summarized in Fig. 5.1): 1) the use of a sparse *patch-based event data representation* that only accounts for the areas of event streams with registered information, and 2) a compact transformer-like backbone based on attention mechanisms [160] that naturally works with this patched information. The proposed transformer-based backbone, in contrast to previous frame-based methods, requires minimal computational resources and makes use of a set of *latent memory vectors* to reduce the computational complexity, but also to encode the seen spatio-temporal event information.

Although both methods share these principles, EvT$^+$ extends EvT in three ways. First, it uses a finer patch-based event data representation with richer spatio-temporal information, while still benefiting from its sparsity. Second, it improves the Event Transformer backbone with a more robust data processing, which we adapt to jointly use information from different data modalities (e.g., event data and grayscale images). In addition to this, we also show how Event Transformer$^+$ can be combined with different output heads to perform either event stream predictions, i.e., classification, or per-pixel predictions, i.e., depth estimation.

We have evaluated EvT and EvT$^+$ in different real event data benchmarks for action and gesture recognition. We have also evaluated EvT$^+$ for dense depth estimation, including a use-case of a multi-modal setup where event data is processed jointly with grayscale images. In all of these tasks, the presented methods achieve better or comparable results to the state-of-the-art, and we show how the improvements from EvT$^+$ outperform EvT for event stream classification. Moreover, we include a thorough analysis of the efficiency and computational complexity of these methods, where we show that both of them are able to work with minimal latency both in GPU and CPU, improving prior work efficiency.

5.2 Related work

This Section summarizes the most common approaches for event data representation as well as event-based Neural Network architectures that process them. It also includes a brief summary of available event-based datasets.

5.2.1 Event data representation

Different from traditional RGB cameras, event cameras log the captured visual information in a sparse and asynchronous manner. Each time an intensity change is detected, the camera triggers an event $e = \{x, y, t, p\}$ defined by its location (x, y) within the space of the sensor grid $(H \times W)$, the timestamp t of the event (in the order of μs) and its polarity p (either positive or negative change).

In order to process the event data, it is first transformed into a representation F. This event representation aggregates the event data $\varepsilon = \{e^1, e^2, e^3, ...\} \mid e^i_t \epsilon \Delta t$ generated during a time-window Δt that covers a time-span from t_i to t_e. These representations can be created differently and, depending on their nature, we divide them into two categories: **event-level representations** usually treat the event data as graphs [165, 14, 15, 34] or point-clouds [140, 161], taking advantage of the event sparsity to achieve better efficiency. Differently, **frame-based representations** group incoming events into dense frame-like arrays, ignoring the event data sparsity but easing a later learning process to achieve better performance. Our work is built on the top of frame-based representations, where we find plenty of variations in the literature. The time-surfaces [77] build frames encoding the last generated event for each pixel. SP-LSTM [111] builds frames where each pixel contains a value related to the existence of an event in a time-window and its polarity. The Surfaces of Active Events [108] builds frames where each pixel contains a measurement of the time between the last observed event and the beginning of the accumulation time. Motion-compensated [122, 162] generate frames by aligning events according to the camera ego-motion. [46] binarizes frame representations in the temporal dimension, achieving a better time resolution. TBR [61] aggregates binarized frame representations into single-bin frames. M-LSTM [19] uses a grid of LSTMs that processes incoming events at each pixel to create a final 2D representation. TORE [9] uses FIFOs to retain the last events for each pixel.

In our solutions (see Fig. 5.1), similarly to previous work [15, 61, 5, 165, 9], we create a frame representation for each time-window. But contrary to prior work, instead of processing the whole frame, we propose to use a **patch-based event data representation** that extracts only the patches (tokens T) of intermediate frame representations with sufficient logged event information. The proposed hybrid solution benefits from both the event-level representations for a later more efficient event data processing since we, to a certain extent, tackle the sparsity of the event data. But we also benefit from the robustness of the frame-based representations to achieve better performance.

5.2.2 Neural Network architectures for event data

Deep Learning-based techniques have shown promising results working with event camera data. This Section discusses the main existing architectures to process different types of event representations, as well as to aggregate the processed information from several time-windows, both for event stream classification and dense predictions. Besides, we provide a short overview of Visual Transformers, since they are actually one of the pillars of the architecture proposed in this work to process events.

Event stream classification has been addressed in different ways in the literature. First, we find some efficient architectures that process sparse event representations such as Spike Neural Networks [67, 145, 172], PointNet-style Networks [165] or Graph neural Networks [14, 15, 34]. Most commonly, others rely on CNNs to process event-frame representations [5, 61, 9, 19]. These methods process whole frames, even if no events are triggered in large parts of them. Despite usually achieving higher accuracy than event-level representations, this unnecessary processing makes frame-based approaches consume more computational resources than needed. For long event stream processing, they are split into shorter time-windows that are frequently processed independently, and then aggregated with Recurrent Networks [61, 168], CNNs [5, 61], temporal buffers [9, 34], or voting between the intermediate results [61].

Depending on the aggregation strategy, we consider that an event-processing algorithm is able to perform **online inference** if it can evaluate the information within each time-window incrementally, as it is generated, and then perform the final visual recognition with minimal latency, as opposed to the processing of all the captured information in a large batch. Our approach performs online inference by updating incrementally a set of latent memory vectors with simple addition operations and processing the resulting vectors with a simple classifier.

Event stream dense estimation is mostly addressed in the literature with dense event-frame representations and CNNs to process them. LMDDE [55] uses fully convolutional networks to process the event data and ConvLSTMs to handle their temporality. ULODE [188] trains a CNN to deblur event representations and predict optical flow, egomotion, and depth. ECN [175] uses an Evenly-Cascaded Convolutional Network to predict optical flow, egomotion, and depth. DTL [164] use CNNs to translate events to images for semantic segmentation and depth estimation. RAM-Net [45] uses CNNs to encode both grayscale and event frames and ConvGRUs to update a hidden state with the temporal information, used later to perform multi-modal depth estimation. LMDDE [55] and RAM-Net [45] propose synthetic datasets to be used as pre-training.

Differently, we complement our sparse patches (i.e., Transformer tokens T), already processed by our backbone, with dummy tokens to create a dense representation that is then updated with the information from our latent memory vectors. Then, similar to RGB solutions, [121] we use skip connections between the self-attention blocks in the encoder and the dense output head to generate the final dense output.

Visual Transformers, initially introduced for Natural Language Processing [160], have recently gained popularity for Visual Recognition tasks. Different from other architectures, they are able to ingest lists of tokens of variable length and process them with attention mechanisms. In other words, contrary to convolutions, attention mechanisms focus on the whole input data (structuring it as a Query (Q), Key (K), and Value (V)) to capture both local and long-range token dependencies:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V. \tag{5.1}$$

Although these models have barely been studied with event data, when it comes to RGB data they can be applied to the features generated by CNNs [22] or, more commonly, to patches (i.e., tokens) extracted from the input images [36, 96]. When working on video data, some methods [7, 97] process patches from different video frames together or aggregate the temporal information with LSTM layers [60]. Of special relevance for this work is Perceiver [64, 63], a transformer that uses latent vectors to process the input data and bound the quadratic complexity of transformers.

In our case, as the patches T (whose length varies on the event data sparsity) from new time-windows are being generated, they are processed by an **attention-based backbone**. This model includes a set of M latent memory vectors that help in this processing, but also these vectors are incrementally updated to encode the information seen so far. The final processing of the latent vectors allows for performing tasks such as event stream classification or dense per-pixel estimations.

5.2.3 Event dataset recordings

Despite their promising applicability, there are still not many large-scale public datasets recorded with these cameras in real scenarios. Therefore, some methods seek the translation of RGB datasets to their event-based counterpart. Earlier event-based solutions [112, 143, 85, 15, 58] display RGB data in an LCD monitor that they record with an event camera. More recent works introduce the use of learning-based emulators [109, 59, 44] to generate event data. Still, these translated datasets cannot fully mimic the event data nature and introduce certain artifacts, especially on their sparsity and latency. This happens because events are triggered by unrealistic lightning conditions and are frequently dependent on the fixed low frame rate of a monitor or the movement of a camera over static images. In order to have a more reliable evaluation setup, we focus our experimentation on datasets recorded with event cameras on real scenarios.

In the case of *event stream classification*, we use two datasets composed of long sequences of between 1 and 6 seconds that contain repetitions of shorter human gestures. The **DVS128 Gesture Dataset** [5] is composed of 1342 event streams capturing 10 different human gestures (plus an optional extra category for random movements) and recorded with 29 different subjects under three different illumination conditions. The **SL-Animals-DVS Dataset** [158] is composed of 1121 event streams capturing 19 different sign language gestures, executed by 58 different subjects, under different illumination conditions. Additionally, we use the **ASL-DVS Dataset** [14] which is composed of shorter gesture sequences of about 100 ms. It contains 100,800 event streams capturing 24 letter signs from the American Sign Language, performed by 5 different subjects. As proposed by the authors, we randomly split the dataset with 80% of the data for training and the remaining 20% for testing.

In the case of *dense estimation*, we use the **MVSEC Dataset** [187] for evaluation. This dataset includes stereo automovilistic recordings with event data, grayscale images, and depth maps recorded by a LiDAR, captured by day (2 recordings) and at night (3 recordings). For our evaluation, we use the depth maps as supervision to perform *dense depth estimation* and, similar to previous work, we use the *outdoor_day_2* sequence for training and the remaining 4 sequences for testing. Due to the corruption of different sequences, we limit the training and validation to the data recorded from the left sensors. Depth maps are always recorded at 20 Hz, but grayscale images are recorded at 45 Hz by day and 10 Hz by night, therefore, they are not synced with the depth maps.

When it comes to the evaluation of the *model efficiency*, since there are no statistics published from prior work on real event stream benchmarks, we use the simulated dataset **N-Caltech-101** [112]. This dataset is the event counterpart of the RGB images from Caltech-101 [38]. Therefore, it is intended for classification and contains short event recordings of the RGB images displayed on an LCD monitor.

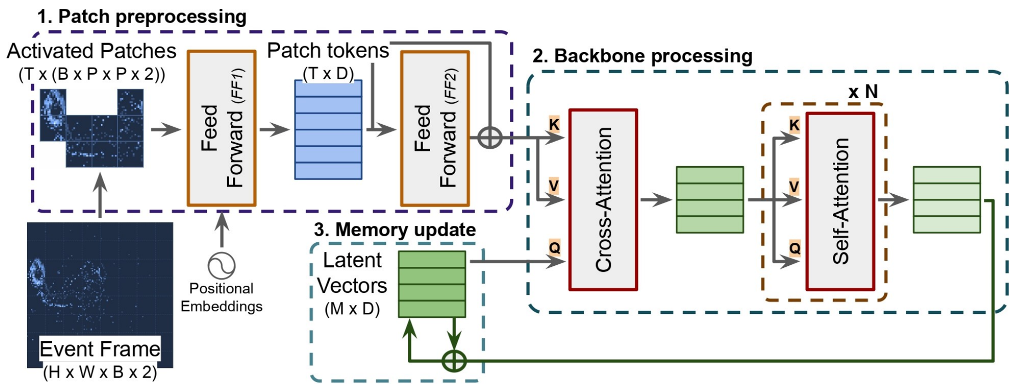
5.3 Event Transformer

This Section describes the EvT framework, presented as an efficient solution for event data processing. In the following, we describe its novelties in terms of event data representation and backbone architecture.

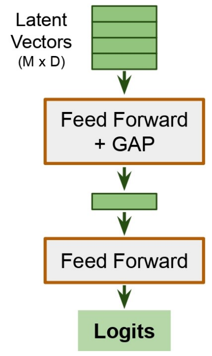
5.3.1 Patch-based event data representation

Similar to previous frame-based methods [61, 5, 9], we transform the events within time-windows Δt into frame representations $F^{H \times W \times B \times 2}$ where each location $(x, y) \mid y \epsilon H, x \epsilon W$ in F contains two histogram-like vectors of B bins, one for each polarity $p \epsilon \{0, 1\}$. Each histogram discretizes Δt in each bin and counts the number of positive or negative events occurring in the corresponding period $\Delta t / B$. Final representations are transformed as $F' = log(F + 1)$ to smooth extremely high values in highly activated areas.

Frame representations are then split into non-overlapping patches of size $P \times P$. Then we set each patch as activated if it contains at least m percent of non-zero elements, i.e., if at least a m percent of the pixels (x, y) within the patch have registered events. Activated patches are kept for further processing by EvT, while the non-activated patches are discarded, reducing significantly the following computation cost and implicitly the ambient noise. If the amount of activated patches is below a threshold n, i.e., there is not enough visual information to be processed, we expand the time-window Δt and increase the event set ε with the new incoming events, recompute the frame representation and extract the activated patches. This last step is repeated until we get at least n activated patches. As a final step, we flatten the T activated patches to create tokens of size $(P^2 \times B \times 2)$, input of the transformer backbone detailed in the next Section.



(a) Backbone overview



(b) Classification architecture

Figure 5.2: Event Transformer (EvT) overview. (a) For each new event frame built from an input event stream, *Activated Patches* are detected and pre-processed to build patch tokens. These tokens are processed by our backbone, and the resulting output is used to update a set of latent vectors. These latent vectors encode all the information received so far. (b) The final version of latent vectors is used to perform the final event stream classification.

5.3.2 Architecture

These activated patches are then processed, as detailed in Figure 5.2, following these steps:

1. Patch pre-processing. Each one of the T input activated patches is mapped to a vector of dimensionality D. This vector length D is constant along the network. This transformation (*FF1*) consists of an initial single-layer Feed Forward Network (FF), the concatenation of 2D-aware positional embeddings, and a last single-layer FF Network. The use of positional embeddings to augment the patch information is required since Transformers, unlike CNNs, cannot implicitly know the locality of the input data. Before being processed by the core of the backbone, transformed patches, i.e., patch tokens, undergo another transformation (*FF2*) composed of a double-layer Feed-Forward Network and a skip-connection to achieve a finer representation while preserving the information about its locality.

2. Backbone processing. The core of the backbone is composed of a single Cross-Attention and N Self-Attention modules that share the same architecture (detailed in Fig. 5.3). Similar to previous transformer-related works [160, 64, 7], it is composed of a Multi-Head Attention layer [160], normalization layers, skip connections and Feed Forward layers. The backbone first processes the latent memory vectors (as Q) on the basis of the patch token information (as $K-V$) (Cross-Attention) and resulting vectors are then refined (as $Q-K-V$) with no external information (Self-Attention).

3. Memory update. Once the patch tokens T and latent vectors have been processed by the backbone, the resulting latent vectors in this iteration are combined with the existing memory latent vectors with a simple sum operation. This augmented version of latent vectors encodes a broader spatio-temporal information and will be used to process

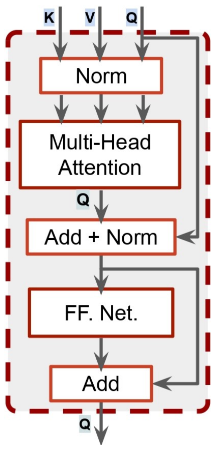


Figure 5.3: Architecture shared by the Cross and Self-Attention modules.

the activated patches from the following time-window.

4. Classification output. The final output of the proposed framework is obtained by processing the latest version of the latent memory vectors, which contain the key spatio-temporal information of the event stream seen so far. In our case, we perform a multi-class classification by simply processing the latent vectors with two Feed Forward Layers and Global Average Pooling (GAP), as detailed in Fig. 5.2b.

5.3.3 Optimization

Event Transformer is optimized for the *event stream classification* task with the Negative Log-Likelihood loss:

$$\mathcal{L}_{NLL} = -\sum_{i}^{n}(Y_i log\widehat{Y}_i + (1-Y_i)log(1-\widehat{Y}_i) \tag{5.2}$$

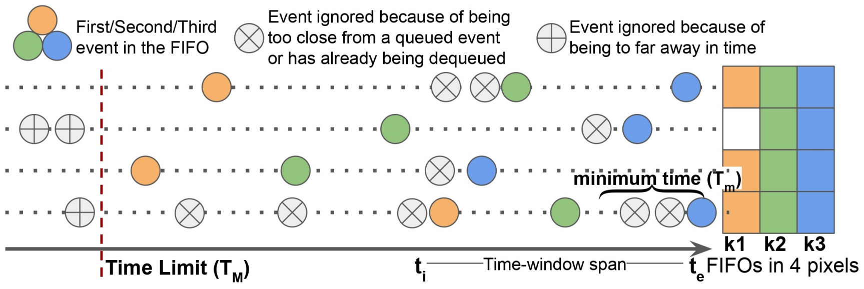
between the predicted labels $\widehat{Y}_i$ and the groundtruth labels Y_i. Besides that, we use label smoothing [178] for regularization.

5.4 Event Transformer$^+$

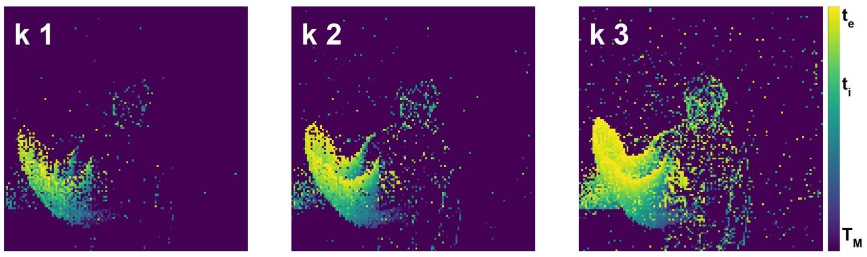
This Section describes the EvT$^+$ framework, built as an extension of Event Transformer. In the following, we describe its improvements in terms of event data representation and backbone architecture.

5.4.1 Patch-based event data representation

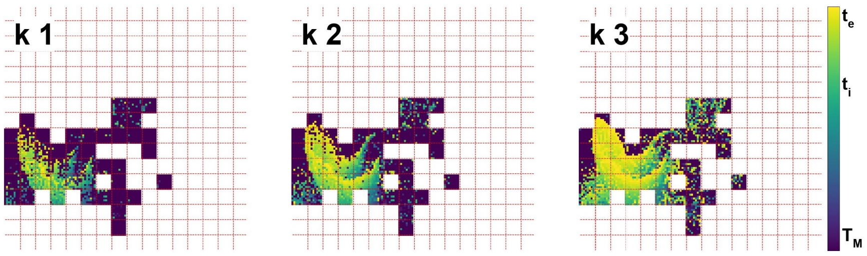
Similar to TORE [9], in EvT$^+$ we model the event information with queues $FIFO(x, y, p, k)$ (see Fig. 5.4a) that retain $k\epsilon K$ events for each pixel $(y\epsilon H, x\epsilon W)$ within the sensor array and polarity $(p\epsilon\{0, 1\})$. But differently, for each pixel, we do not retain the last K events but the last K events that are separated by at least a minimum time of $T_m = \frac{\Delta t}{K}$. This threshold is intended to avoid the over-representation of the information provided by the events that happen consecutively in time. Additionally, when no events are registered in this time-window span for a certain pixel, we account for the ones triggered up to a maximum time T_M ($T_M \gg \Delta t$ and $T_M \ll t_i$).


First/Second/Third event in the FIFO
Event ignored because of being too close from a queued event or has already being dequeued
Event ignored because of being to far away in time
minimum time (T_m)
k1 k2 k3
Time Limit (T_M)
t_i Time-window span t_e
FIFOs in 4 pixels

(a) Event accumulation using FIFOs of size $K = 3$



(b) Sample frame-like representation for different values of K



(c) Activated patches for different values of K

Figure 5.4: Patch-based event data representation. (a) For each pixel, we retain the last K events taking into account sufficient sparsity in time. (b) Frame-like representations are built with the timestamps of the queued events. (c) Frame representations are split into patches, keeping only the activated patches, i.e., with enough event information.

Once the events are queued for a given time-window, we build the intermediate frame representation $F^{H \times W \times K \times 2}$ with their timestamps (see Fig. 5.4b). We normalize the pixels to have a value in the range from 0 to T_M and then scale their values to a $0-1$ range (Eq. 5.3):

$$F = F - (t_e - T_M), \; F = F/T_M \tag{5.3}$$

Therefore, the events queued at the end of the time-window will have values close to 1 and the ones close to T_M will have a value close to 0.

Then, similar to EvT, we split the generated frame-representations into non-overlapping patches of size $P \times P$ (see Fig. 5.4c), and we set each patch as *activated* if it contains a minimum m percent of pixels that have information of events triggered between t_i and t_e. Note that events triggered between T_M and t_i are not involved in the patch activation

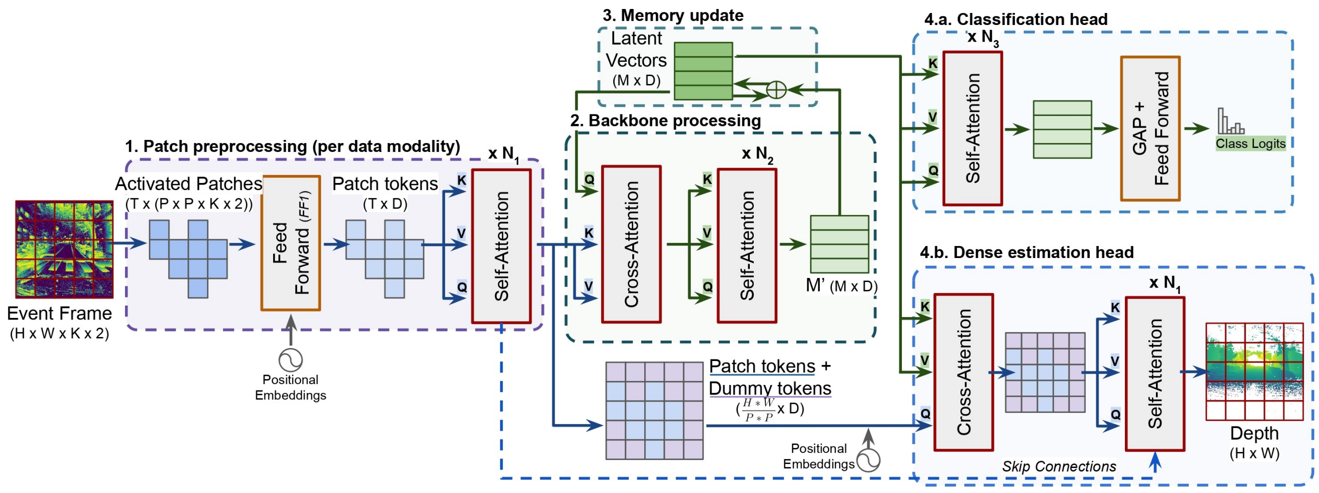


Figure 5.5: Event Transformer$^+$ (EvT$^+$) overview. The set of patch tokens T related to the events within a time-window (or any other data modality such as images) is processed to update a set of latent memory vectors. The latter accumulates the seen information and is used to perform the final downstream task, i.e., in our framework event stream classification or dense per-pixel estimation.

decision, since they have been considered in previous time-windows, but they complement the patch information to ease later their processing. Activated patches are finally flattened to create tokens T of size $(P^2 \cdot K \cdot 2)$, input of the transformer backbone detailed in the next subsection.

5.4.2 Architecture

These activated patches are then processed, as detailed in Figure 5.5, following these steps:

1. Patch pre-processing Each one of the T input activated patches is mapped to a vector of dimensionality D, constant along the network. This transformation (*FF1*) consists of an initial Feed Forward layer (FF), the concatenation of 2D-aware positional embeddings, and a last FF layer. An initial set of N_1 Self-Attention blocks is then used to analyze long and short-range spatial dependencies between tokens. In the case of *multi-modal data processing*, a different patch pre-processing branch is used for each data modality.

2. Backbone processing. This backbone uses the pre-processed token information (as $K - V$) to process the latent memory vectors M (as Q) with a single Cross-Attention Module. The resulting M' vectors are then refined with N_2 Self-Attention layers.

3. Memory update. The latent memory vectors M are updated given the newly generated vectors M' with a simple sum operation and normalization:

$$M = \|M + M'\| \tag{5.4}$$

This augmented version of the latent vectors encodes longer spatio-temporal information and is used to perform the final downstream task, for which we have implemented the following two options.

4.a. Classification head. Event stream classification is performed by processing the refined latent memory vectors, that contain the key spatio-temporal information of the event stream seen so far. This processing consists of N_3 Self-Attention modules and then, similar to EvT, processing the resulting vectors with two Feed Forward layers and Global Average Pooling (GAP).

4.b. Dense estimation head. Given the sparse set of tokens processed at step 1 and their initial location in the input frame representation, we convert them back to a dense representation by adding dummy tokens (initialized with zeroes) that complement the ones lost due to the event data sparsity. We then add positional embedding information to this dense representation and update it with the information contained in the latent memory vectors (used as $K - V$) with a Cross-Attention layer. The resulting final set of tokens is then refined with N_1 Self-Attention layers, which have skip-connections from the N_1 Self-Attention layers of the patch pre-processing step.

In the case of *multi-modal data processing*, the skip connections propagate the information jointly for each data modality, whose tokens are merged with a simple addition and normalization operation.

Attention modules. All the Cross and Self-Attention modules from EvT$^+$ share the same architecture, similar to EvT and previous transformer-related works [160, 64, 7], composed of a Multi-Head Attention layer [160], normalization layers, skip connections and Feed Forward layers.

5.4.3 Optimization

Event Transformer$^+$ is optimized differently for the different downstream tasks. In the case of *event stream classification*, we optimize EvT$^+$ in the same way as EvT (see Section 5.3.3) with the Negative Log-Likelihood loss (equation 5.2).

In the case of *monocular dense estimation*, we train EvT$^+$ in the sparse depth labels measured by a LiDAR sensor. These groundtruth depth maps Y, similar to other methods [45, 55], are clipped to a range $[D_m - D_M]$ captured by the sensor ($[2 - 80]$ in our case of MVSEC Dataset) and we train EvT$^+$ to predict its normalized log depth representation $\bar{Y}\epsilon[0 - 1]$:

$$\bar{Y} = \frac{log(Y) - log(D_m)}{log(D_M) - log(D_m)}. \tag{5.5}$$

We optimize EvT$^+$as in [45] with a scale-invariant loss

$$\mathcal{L}_{si} = \frac{1}{n}\sum_i^n (R_i)^2 - \frac{1}{n^2}(\sum_i^n R_i)^2, \tag{5.6}$$

and a multi-scale invariant loss

$$\mathcal{L}_{msi} = \frac{1}{n}\sum_k^4 \sum_i^n (|\nabla_x R_i^k|| + ||\nabla_y R_i^k||), \tag{5.7}$$

where n are the valid depth groundtruth points, R_i is the log-depth difference map $||\bar{Y_i} - \widehat{Y_i}||$ at the point i between the groundtruth $\bar{Y_i}$ and the predicted $\widehat{Y_i}$ log-normalized depth, R_i^k is the log-depth difference map at the scale $k\epsilon[0 - 4]$. Both losses are combined as $\mathcal{L} = \mathcal{L}_{si} + \lambda\mathcal{L}_{msi}$, with $\lambda = 0.25$.

5.5 Experiments

Following Sections include the evaluation results of the two presented frameworks for event stream classification and dense estimation. Next, we analyze the efficiency of the presented models and detail their implementation and training details, as well as the set of ablation experiments performed to take those design choices.

5.5.1 Event stream classification evaluation

The evaluation for event stream classification, i.e., action and gesture recognition, is performed in two different kinds of datasets. First kind contains long sequences that present longer and more complex temporal contexts. Second kind contains shorter sequences with much simpler motion and temporal complexity.

The evaluation for **long event stream classification** includes two different datasets, DVS128 Gesture Dataset [5] and SL-Animals-DVS Dataset [158] (details in Section 5.2.3). To fit our computing resource restrictions, sequences from these datasets are cropped temporally (details in Section 5.5.4) and split into time-windows Δt of 24 and 48 ms for the DVS128 and SL-Animals-DVS datasets respectively.

Table 5.1 shows the accuracy of top-performing models in the DVS128 Dataset, with and without including the extra additional distractor class of random movements (11 and 10 classes respectively). The column *Online* highlights the ability of each model to perform online inference, i.e., incremental processing of the event data and classification with low latency. Similarly, Table 5.2 shows the accuracy of top-performing methods evaluated on the SL-Animals-DVS Dataset, a more demanding benchmark with lower state-of-the-art accuracy. *3 Sets* results exclude the samples recorded indoors with artificial lighting from a neon light source since they include noise related to the reflection of clothing and the flickering of the fluorescent lamps. *4 Sets* evaluates all the samples within the dataset.

Model	**10 Classes**	**11 Classes**	**Online**
RG-CNN [15]	N/A	97.2	x
3D-CNN + Voting [61]	**99.58**	**99.62**	x
CNN [5]	96.49	94.59	✓
Space-time clouds [165]	97.08	95.32	✓
CNN + LSTM [61]	97.5	97.53	✓
TORE [9]	N/A	96.2	✓
EvT	98.46	96.20	✓
EvT$^+$	**99.24**	**97.57**	✓

Table 5.1: Classification Accuracy in DVS128 Gesture Dataset. N/A = Not Available at the source reference.

Results from Table 5.1 show how EvT obtains comparable results to the state-of-the-art and EvT$^+$ manages to outperform prior work on both data setups. Only [61] is more accurate than EvT$^+$ but it uses offline inference and 3D-CNNs, which are computationally expensive but have a good inductive bias, useful when training with small datasets like DVS128 and with random movements (as it is the case of 11 Classes). As for the more challenging SL-Animals Dataset, both EvT and EvT$^+$ manages to outperform prior work. Interestingly, our solutions present higher robustness to the different lighting conditions

Model	**3 Sets**	**4 Sets**
SLAYER [159]	78.03	60.09
STBP [159]	71.45	56.20
DECOLLE [67]	77.6	70.6
TORE [9]	N/A	85.1
EvT	87.45	88.12
EvT$^+$	**92.34**	**94.39**

Table 5.2: Classification Accuracy in SL-Animals-DVS. N/A = Not Available at the source reference.

of the *4 Sets*, being able to take advantage of larger training set to achieve better accuracy. Moreover, the improvements introduced in EvT$^+$ help to outperform the results from EvT in both datasets by a large margin.

In the case of **short event stream classification**, we evaluate EvT in the ASL-DVS Dataset (details in Section 5.2.3). As many other methods that tackle this problem [14, 19, 34, 9], we treat it as a simple instance classification problem where a single event representation (from a single time-window Δt) is built from the whole event stream.

Model	**Accuracy**
RG-CNN [14]	90.1
EV-VGCNN [34]	98.3
M-LSTM [19]	99.73
TORE [9]	99.6
EvT	**99.93**

Table 5.3: Classification Accuracy in ASL-DVS (short event streams).

To perform this short event stream classification, we represent each event stream with a single frame representation that encodes $100ms$ of the sample ($\Delta t = 100ms, B = 2$ for EvT). Then, the final classification is performed by processing the generated activated patches with the EvT backbone and computing the logits from the output latent vectors, with no need to perform any memory update. Table 5.3 shows that EvT is able to get excellent performance. Note that, since this experiment does not present a high complexity, we do not evaluate EvT$^+$ for short-event stream classification.

5.5.2 Dense estimation evaluation

Since EvT does not have a dense estimation head, this experiment is only intended to evaluate EvT$^+$. In particular, we evaluate monocular depth estimation on the MVSEC Dataset [187] (details in Section 5.2.3). Although the recorded sequences are very long, due to computational restrictions, we just consider (both for training and validation) the 512 ms of information for the depth map estimation. The event information from these sequences is split in time-windows Δt of 50 ms that are synced with the depth maps and is complemented with a grayscale image generated at least $\Delta t/2$ ms away from the end of each time-window. Therefore, all time steps contain event data but might not contain grayscale image information. This issue is more frequent in the night sequences, where the grayscale image frequency is lower. When there is no grayscale information, only the event tokens update the memory and are used for the later dense depth estimation.

Table 5.4 shows the average depth error of different models at different cut-off depths, i.e., pixels whose groundtruth depth information is under the specified threshold (10, 20, or 30 meters). As observed, EvT$^+$ is able to largely outperform previous methods in most of the setups, with no specific pre-training. More importantly, when including image data EvT$^+$ improves its accuracy to achieve higher robustness even in the most challenging scenarios.

Model	**outdoor day 1**			**outdoor night 1**			**outdoor night 2**			**outdoor night 3**		
	10	**20**	**30**	**10**	**20**	**30**	**10**	**20**	**30**	**10**	**20**	**30**
	Event data processing											
ULODE [188]	2.72	3.84	4.40	3.13	4.02	4.89	2.19	3.15	3.92	2.86	4.46	5.05
LMDDE [55]	2.70	3.46	3.84	5.36	5.32	5.40	2.80	3.28	3.74	2.39	2.88	3.39
LMDDE [55]*	1.85	2.64	3.13	3.38	3.82	4.46	1.67	2.63	3.58	**1.42**	2.33	3.18
EvT$^+$(Ours)	**1.29**	**2.04**	**2.46**	**1.68**	**2.21**	**2.82**	**1.63**	**2.00**	**2.95**	1.53	**2.20**	**2.79**
	Multi-modal (events + images) processing											
RAM Net [45]†	1.39	2.17	2.76	2.50	3.19	3.82	**1.21**	2.31	3.28	**1.01**	2.34	3.43
EvT$^+$(Ours)	**1.22**	**1.85**	**2.26**	**1.52**	**2.04**	**2.77**	1.50	**2.15**	**2.88**	1.30	**2.04**	**2.76**

* Pre-training on DENSE[55] dataset. † Pre-training on EventScape[45] dataset.

Table 5.4: Evaluation on the MVSEC Dataset. Average absolute depth error in meters (lower is better) at different cut-off depth distances in meters (10, 20, 30). First block shows models trained just on event data. Second block shows models trained jointly with event and image (grayscale) data.

5.5.3 Efficiency analysis

We now provide a deeper analysis of the computational cost of the presented frameworks, and how they benefit from the event data sparsity. The **computational cost of EvT** is determined by the Cross-Attention Layer, which has $O(|T| \times |M|)$, where $|T|$ stands for the amount of activated patches and $|M|$ for the amount of latent memory vectors. Therefore, the fewer activated patches there are, the fewer resources EvT needs. But also, in the worst case, when many activated patches are found ($|T| \gg |M|$), the latent vectors prevent EvT from incurring a quadratic cost. Note that the Self-Attention layers have then a reduced cost of $O(M^2)$ instead of $O(|T|^2)$. Differently, **the computational cost of EvT$^+$**, in the case of event stream classification, depends on the initial self-attention pre-processing ($O(|T|^2)$). This means that, similar to EvT, the cost lowers as the input data is more sparse, but it is not bounded by $|M|$. In the case of depth estimation, the existence of a dense output head that does not work with sparse information increases the computational cost to $O(\frac{H*W}{P*P}^2)$. Still, the rest of the Neural Network benefits from the data sparsity by suppressing non-activated patches. This is also true for the ones generated from grayscale images, when pixels are black, especially in the night sequences.

Unfortunately, there are no **model efficiency** statistics published from prior work on real event stream benchmarks. Therefore, we use the simulated dataset N-Caltech-101 [112] (details in Section 5.2.3) to compare the efficiency of EvT with prior work (Table 5.5), and then we calculate and compare the efficiency statistics of EvT and EvT$^+$ in the rest of datasets evaluated in this Chapter (Table 5.6).

Table 5.5 shows the model complexity (measured in FLOPs and number of model parameters, which is directly related to energy consumption) required by different methods evaluated on the N-Caltech-101 dataset. As observed, EvT presents much lower requirements than frame-based methods, that run heavy computations. More importantly, EvT also presents lower computational requirements than point-based methods, which are also designed to tackle the sparsity of the event data but achieve less computational savings than our approach.

Model	**Type**	**(G)FLOPs**	**#Params.**
RG-CNN [14]	Point-based	0.79	N/A
EV-VGCNN [34]	Point-based	0.70	0.84 M
M-LSTM [19]	Frame-based	4.82	21.43 M
EvT	**Patch-based**	**0.20**	**0.48 M**

Table 5.5: Average FLOPs for all validation samples in N-Caltech-101 and number of parameters per model. N/A = Not Available at the source reference

Table 5.6 shows more **detailed efficiency statistics** of EvT and EvT$^+$ in the rest of the datasets evaluated in this Chapter. As observed, for short event stream datasets (N-Caltech-101 and ASL), EvT generate many patches ($T \gg M$) since they are recorded with a bigger sensor (240×180 pixels) and, in the case of N-Caltech-101, because of its synthetic nature. Otherwise, long event stream datasets (Sl-Animals and DVS128) generate fewer patches ($T < M$) per time-window since they use a smaller sensor (128×128 pixels) and shorter time-windows and, in the case of EvT$^+$, also because of using a bigger patch size P. Although the amount of activated patches $|\mathbf{T}|$ of the first group has an order of magnitude more than the second, we do not observe this trend in its computational cost (both in time and FLOPs), which grows smoothly with $|\mathbf{T}|$. Moreover, although the computational cost of EvT$^+$ is theoretically higher than the one of EvT, different implementation improvements such as bigger patch sizes P and less latent vectors M, make EvT$^+$ even more efficient (FLOPs and latency) than EvT for event stream classification.

Model	**Sensor size**	**Dataset**	$\mathbf{\|T\|}$	$\mathbf{\Delta t}$ **ms**	**Latency (GPU/CPU)**	**FLOPs**	**#Params.**
EvT	240×180	N-Caltech-101	532	100	4 / 16 ms	0.20 G	0.48 M
EvT	240×180	ASL	263	100	4 / 9 ms	0.13 G	0.48 M
EvT	128×128	SL-Animals	80	48	3 / 5 ms	0.09 G	0.48 M
EvT	128×128	DVS128	45	24	2 / 4 ms	0.08 G	0.48 M
EvT$^+$	128×128	SL-Animals	32	48	2 / 4 ms	0.04 G	0.66 M
EvT$^+$	128×128	DVS128	18	24	3 / 3 ms	0.03 G	0.66 M
EvT$^+$	346×260	MVSEC	318	50	10 / 25 ms	2.94 G	1.98 M
EvT$^+$ (multi-modal)	$346 \times 260 \times 2$	MVSEC	318 + 319	50	15 / 37 ms	3.68 G	2.50 M

$|\mathbf{T}|$: Amount of patch tokens — $\mathbf{\Delta t}$: time-window length
GPU: NVIDIA GeForce RTX 2080 Ti — **CPU**: Intel Core i7-9700K

Table 5.6: EvT$^+$ efficiency analysis: execution time and FLOPs per Δt. Average results for all validation samples in each dataset.

When it comes to dense estimation, since the computational cost of EvT$^+$ is quadratic and the evaluation dataset uses a bigger sensor size that generates more activated patches, EvT$^+$ has a higher cost (FLOPs and parameters) than for event stream classification. In particular, as observed in Table 5.7, the most costly part of the network is the output head, which does not benefit from the event data sparsity.

EvT$^+$ Step	**FLOPs**
Sparse token pre-processing	0.61
Backbone processing and Latent Vectors update	0.06
Dense output head	2.27

Table 5.7: Average FLOPs required to process a single time-window Δt and generate a dense output for depth estimation.

It is important to remark that, since our networks are very shallow, in all cases both EvT and EvT$^+$ processes the activated patches in a significantly shorter time span (see Table 5.6) than the corresponding time-window Δt, both in GPU and CPU. This highlights the ability of our methods to do inference with minimal latency, being capable of processing the event data before the following batch (i.e., time-window data) is generated, therefore facilitating an online inference, even in low-resource environments.

Although we do not have efficiency statistics of prior work evaluated in non-simulated event datasets, we can observe that they are mostly based on heavy computation algorithms. Both for event stream classification and dense estimation, we find that the best-performing methods are dense models based on CNNs [5, 61, 9, 188, 55, 45] that process frame-event representations and include complex aggregation architectures such as Recurrent Layers [61, 19, 55, 45] or CNNs [15] to process intermediate time-window results. Differently, our approaches process event representations with minimal cost benefiting from their sparsity (see Table 5.5 for a similar comparison in the next dataset), and incrementally aggregate intermediate results on the latent vectors used for the final classification with negligible cost. Given this, both EvT and EvT$^+$ are able to perform online inference, while being significantly more resource-efficient than the related methods.

5.5.4 Event Transformer implementation details and ablation

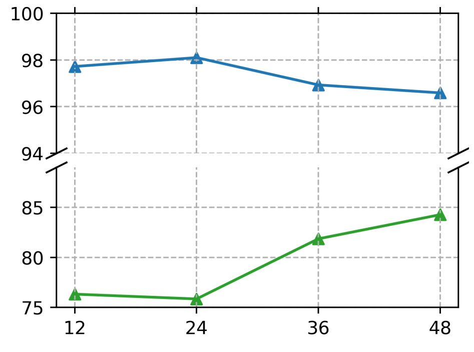
This subsection first resumes the implementation and training details of EvT, and then analyzes and justifies these design choices with detailed ablation experiments on both the DVS128 Gesture (10 classes) and the SL-Animals-DVS (4 sets) datasets.

Implementation details. For the **patch-based event representation** we set a common patch size of 6×6 pixels, but specific values for time-window Δt and number of bins B are specific for each dataset and defined in the following. For all cases, to consider a patch as *activated* we set $m = 7.5$, i.e., we require a 7.5% of the pixels within the patch to log events. Besides, we set $n = 16$, i.e., an event frame must have at least 16 active patches to continue the processing. As previously detailed, if the frame does not have enough ($\geq n$) active patches we increase the time-window covered by the frame and repeat the search for active patches.

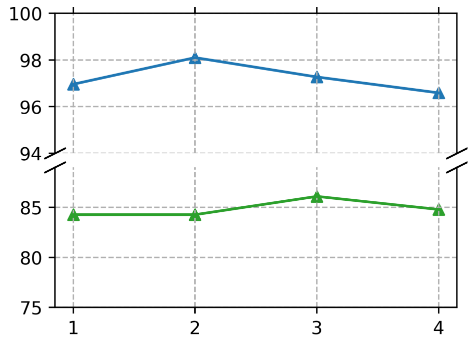
As for the **backbone hyperparameters**, the dimensionality D of the latent vectors and the pre-processed patch tokens is set to 128. The latent memory is composed of 96 latent vectors randomly initialized at the beginning of the training with a Normal distribution of mean 0.0 and deviation 0.2. Similarly, the positional encodings are initialized with 16 bands of 2D Fourier Features [153] (dimensionality of $\frac{H}{P} \times \frac{W}{P} \times 64$, being H and W the specific sensor height and width from each dataset). Both the latent vectors and positional encodings are learned as the rest of the parameters of the network during training. Patch tokens are processed by a single Multi-Head Cross-Attention layer and 2 Multi-Head Self-Attention layers, all of them using 4 attention heads.

Training details. The whole framework is optimized with a Negative Log Likelihood and AdamW [99] optimizer, in a single NVIDIA Tesla V100. The initial learning rate is set as $1e-3$, and we use Stochastic Weight Averaging [62] and gradient clipping. Due to computation resource constraints, we perform the ablation experiments with a batch size of 64 and we reduce the learning rate by a factor of 0.5 after 10 epochs with no loss reduction. Differently, the benchmark results are obtained using a batch size of 128 and a *1cycle learning rate* policy [146] for 240 epochs. As for the data augmentation, we use spatial and temporal random cropping, dropout, drop token, and we repeat each sample within the training batch twice with different augmentations.

Ablation of event representation hyperparameters. The key components of our event representation are the time-window Δt used to aggregate events into patch representations, and the number of bins B used to improve their time resolution. Figure 5.6a shows the EvT accuracy using different time-window lengths Δt (and bins $B = 2$). As observed, the best Δt value is dependent on the evaluated dataset, requiring longer time-windows for SL-Animals-DVS (48 ms) than for DVS128 (24 ms); these time-window values are now set as default for these datasets. Intuitively, shorter time-windows are required to model motions that are executed at higher speeds, and longer ones fit better slower motions that register less event information. Figure 5.6b shows that using more bins B (3) is beneficial when using longer time-windows (SL-Animals-DVS) and using fewer bins B (2) helps with shorter time-windows (DVS128); these bins values are set now as default for these datasets. As observed, the use of more bins helps us to use finer time resolutions and therefore improve the final classification accuracy. However, too many bins represent very short time intervals that do not contain representative event information.



(a) Time-window Δt (ms)



(b) Bins B

Figure 5.6: EvT accuracy with different time-windows Δt and bins B for the DVS128 Dataset - 10 classes (top blue line) and SL-Animals-Dataset - 4 Sets (bottom green line).

Regarding the generation of active patches from event representations, the patch size P is the most relevant hyperparameter, defining the granularity of the information we are working with. As observed in Figure 5.7, smaller patch sizes generate more active patches, making EvT process more information, while bigger patches speed up the EvT inference, but provide less spatial information.

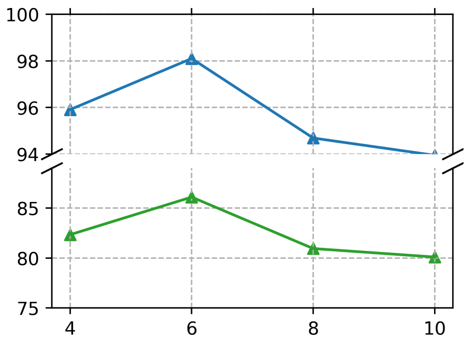


Figure 5.7: EvT accuracy with different patch size value for the DVS128 Dataset - 10 classes (top blue line) and for the SL-Animals-Dataset - 4 Sets (bottom green line).

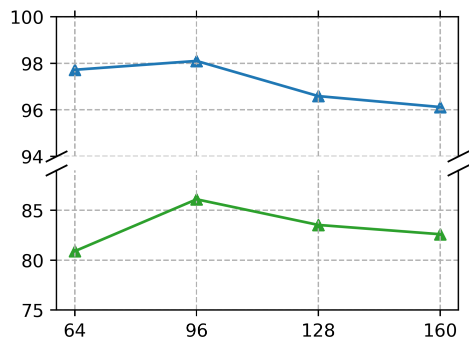
Besides the patch size, the generation of active patches also depends on the minimum amount of pixels m with logged events needed to activate a patch, and the minimum amount of patches n required to start the EvT backbone processing. Table 5.8 shows how filtering out patches with less event information improves the final accuracy, mainly because it removes the patches that contain noisy event information, not related to the motion in the scene itself. Additionally, setting a minimum amount of activated patches helps to avoid the processing of time intervals with no motion and therefore with few activated patches.

n \ m	5%	7.5%	10%
8	97.6 84.6	97.9 83.9	96.6 83.3
16	96.9 85.9	98.1 86.1	98.0 84.3
24	97.2 84.3	97.2 82.4	96.9 82.0

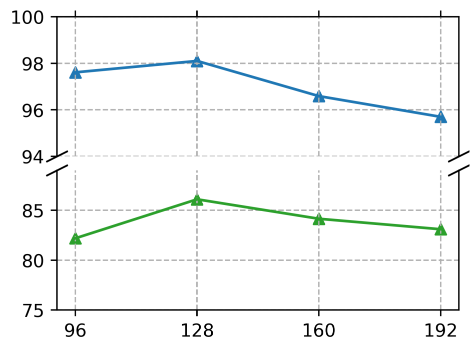
Table 5.8: EvT accuracy with different combinations of m: min. pixels per patch and n: min. patches per Δt. Top cell value: DVS128 (10 classes). Bottom cell value: SL-Animals (4 Sets).

Ablation of Transformer hyperparameters. The number of latent vectors is key for a good generalization. Figure 5.8a shows that using too few or too many vectors causes under or overfitting, leading to suboptimal results. The latent vectors dimensionality (same dimension as the patch tokens) also affects the results. As observed in Fig. 5.8b, too short lengths lead to underfitting results, and too long lengths lead to unstable learning that ends with inaccurate results.

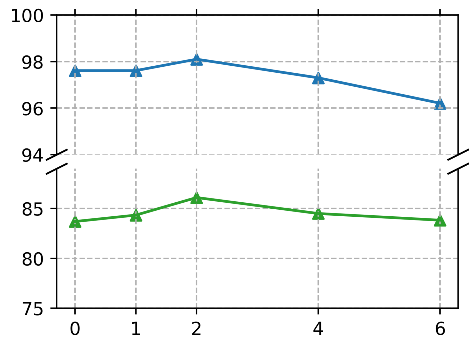
As for the attention hyperparameters, Figure 5.8c shows how the Self-Attention allows better processing of the patch tokens up to a limit where the accuracy is no longer improved. Similarly, Fig. 5.8d shows how more attention heads allow to have finer processing of the input patch tokens, but it can lead to unstable learning and over-fitting.



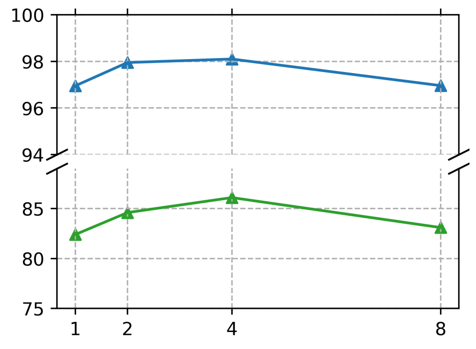
(a) Number of Latent Vectors



(b) Embedding dimension



(c) Number of Self-attention Layers



(d) Number of Attention Heads

Figure 5.8: EvT accuracy for different architecture variations using the DVS128 Dataset - 10 classes (top blue line) and the SL-Animals-Dataset - 4 Sets (bottom green line).

Explainability. The analysis of the attention scores generated at the Multi-Head Attention layers allows to understand which data features (patch tokens) lead to classification decisions. Figure 5.9 shows these attention scores calculated from samples of different datasets. See the supplementary video[2] for more detailed visualization of the attention scores.

5.5.5 Event Transformer$^+$ implementation details and ablation

This subsection first resumes the implementation and training details of EvT$^+$, and then analyzes and justifies the main design choices with detailed ablation experiments on both the DVS128 Gesture (10 classes) and the SL-Animals-DVS (4 sets) datasets. It also discusses the hyperparameters that are key to achieving better results for monocular depth estimation. For the latter study, we use the multimodal version of EvT$^+$.

[2]Supplementary video available at: `https://github.com/AlbertoSabater/EventTransformer`

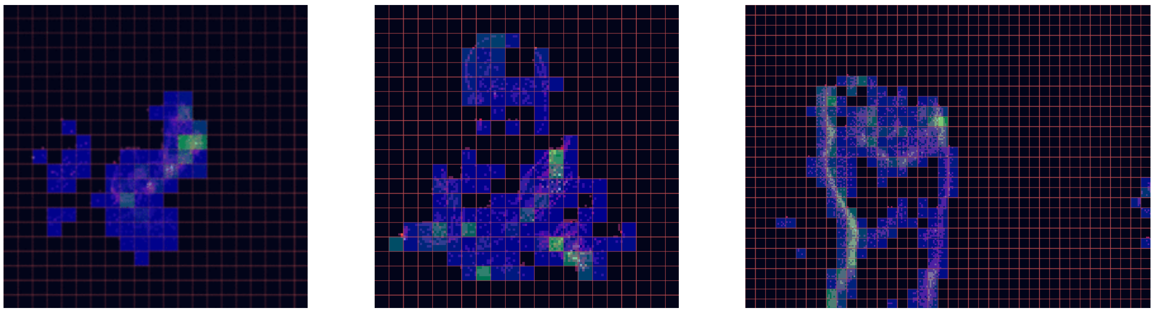

(a) DVS128 Dataset (b) SL-Animals-Dataset (c) ASL-DVS Dataset

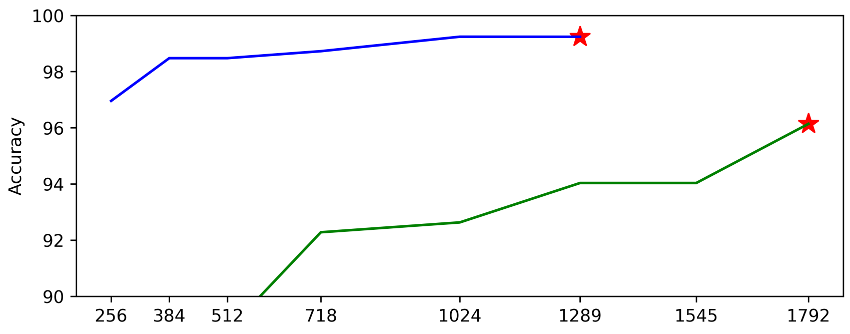
Figure 5.9: Attention scores calculated in one of the heads of the Cross-Attention layer from EvT when processing a sample event stream of each dataset. Greener patches have a higher influence than bluer ones for the final classification decision. Black patches are ignored in the sequence processing.

Implementation details. For the **patch-based event representation** we set a patch size of 10×10 for event stream classification and 12×12 for the depth estimation task, which has larger frame representations. In all cases the number of events in the FIFO, K, is set to 3, M_T is set to 256 ms, and the threshold m for the patch activation is set as in EvT (7.5%). Dataset-specific hyperparameters are discussed in the following.

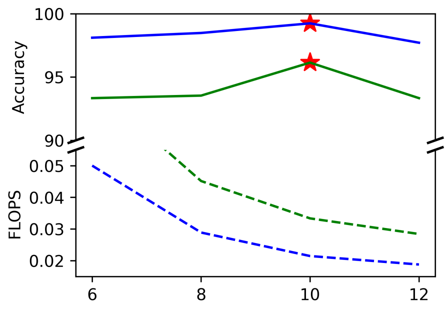
As for the **backbone hyperparameters**, the latent vectors and the vector dimensionality D are set to 160. The latent memory is composed of 32 latent vectors. The positional encodings are initialized with 6 bands of 2D Fourier Features [153] ($\frac{H}{P} \times \frac{W}{P} \times 24$, being H and W the specific sensor height and width from each dataset). Both the latent vectors and positional encodings are learned as the rest of the parameters of the network during training. The number of Attention layers N_1, N_2, and N_3 are set to 1, but in the case of depth estimation, N_1 is set to 2. All the Multi-Head Attention layers use 8 heads, but in the case of depth estimation that has a bigger input size, we use only 4 in the pre-processing and decoding steps to increase their efficiency.

Training details. The whole framework is optimized with the AdamW optimizer [99] in a single NVIDIA Tesla V100, with the learning rate set to $1e-3$ and using gradient clipping. The batch size is 128 for event stream classification and 24 for depth estimation. Data augmentation used consists of spatial and temporal random cropping, dropout, drop token, and repetition of each sample within the training batch twice with different augmentations.

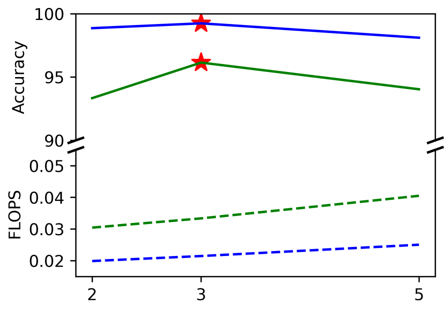
Ablation of event representation hyperparameters. The key components of our event representation are the patch size (P) used to split the event frame representations, the number of events K queued for each FIFO, and the maximum sequence length used to represent each event stream sequence. As observed in Fig. 5.10b, a higher patch size reduces the floating point operations required to process a single time-window, but from some point, it also reduces the model accuracy. Differently, a higher value of the parameter $\underline{K}$ (Fig. 5.10c), used to split the event information for a certain time-window, also increases the required FLOPs without a significant increment of the model accuracy. In both cases, we select, as observed in the Figures, the value that presents a good trade-off between computational cost and accuracy. Finally, we observe in Fig. 5.10a that, in general terms, the accuracy increases with the sequence length used to describe an event stream. It is important to notice that this length converges to maximum accuracy since the sequences found in the datasets are made out of the repetition of shorter action movements. Also, larger event stream sequence lengths could not be tested because of GPU memory restrictions.



(a) Sequence Length



(b) Patch Size (P)

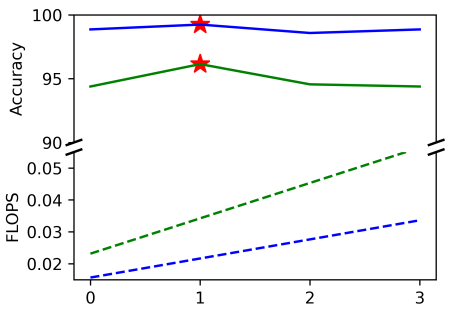


(c) K

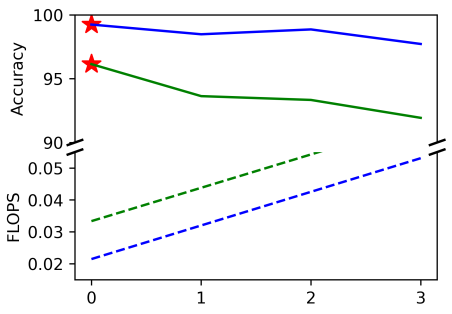
Figure 5.10: EvT$^+$accuracy with different data hyperparameters for the DVS128 Dataset - 10 classes (blue) and for the SL-Animals-Dataset - 4 Sets (green). Dashed lines: average FLOPs required to process a single time-window. Stars: the selected hyperparameter value.

Ablation of backbone hyperparameters. The most relevant backbone hyperparameter for the computational cost is the amount of self-attention layers applied over the input patch tokens. As observed in Fig. 5.11a, its use increases the model accuracy, but the application of many of these layers also increases the model complexity, without a benefit over its accuracy. This is probably due to an increase in model instability during training. As observed in Fig. 5.11b, the use of self-attention after the cross-attention layers (Fig. 5.11c) does not benefit EvT$^+$, both in accuracy and model complexity. Differently, using self-attention in the classification decoder (Fig. 5.11d) helps the classification accuracy with minimal overhead. Finally, we note that different from EvT, using more latent vectors does not help the training accuracy.

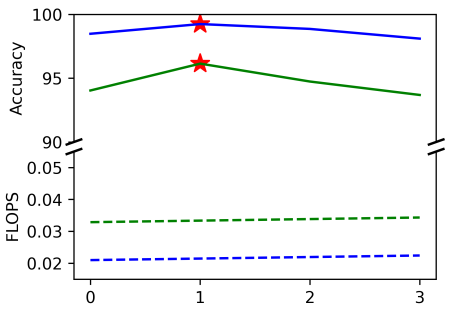
Ablation of backbone hyperparameters for dense estimation. For multi-modal dense estimation where the sensor used to record the sequences have a bigger size, we find that increasing the patch size (Fig. 5.12a) not only reduces the FLOPs but also increments the model performance (lower mean error). Moreover, since this data presents higher



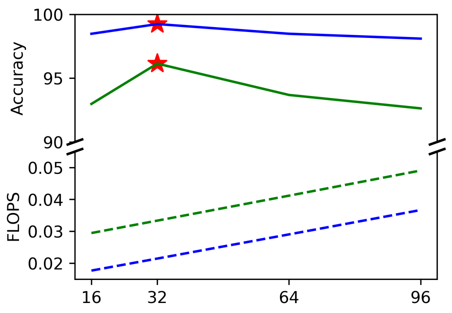
(a) Pre-proc. Self-Attention Blocks ($N1$)



(b) Backbone Self-Attention Blocks ($N2$)



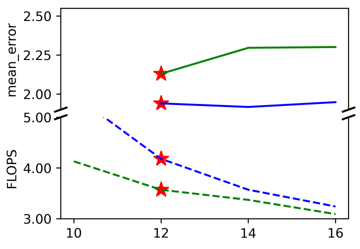
(c) Decoder Self-Attention Blocks ($N3$)



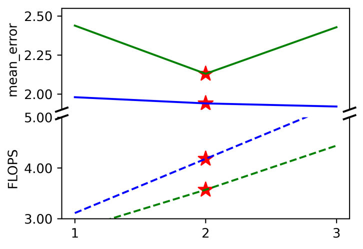
(d) Latent Vectors (M)

Figure 5.11: EvT^{+}accuracy with different model hyperparameters for the DVS128 Dataset - 10 classes (blue line) and for the SL-Animals-Dataset - 4 Sets (green line). Dashed lines: average FLOPs operations required. Stars: the selected hyperparameter value.

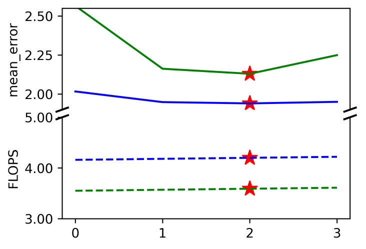
complexity, we find that a more complex network is required to achieve better results. In particular, we find that the amount of pre-processing and backbone Self-Attention blocks (Fig. 5.12b and 5.12c) significantly affects the model performance, but at the cost of augmenting the computational requirements.



(a) Patch Size P



(b) Pre-proc. Self-Attention Blocks ($N1$)



(c) Backbone Self-Attention Blocks ($N2$)

Figure 5.12: EvT^{+}average depth error (20m crop rate) and FLOPs required to process a single time-window of the *outdoor_day_1* (blue) and *outdoor_night_2* (green) MVSEC sequences. Lower is better. Stars: selected hyperparameter value.

5.6 Conclusions

This Chapter introduces two novel frameworks for event data processing, Event Transformer and its updated version, Event Transformer$^+$. These two works effectively manage to take advantage of event data properties to minimize its computation and resource requirements, while achieving top-performing results. This is achieved with our proposed patch-based event representation that only accounts for parts of the streams with sufficient logged information, and an efficient and compact transformer-like architecture that naturally processes it. The proposed work achieves state-of-the-art results for action and gesture recognition and, in the case of EvT$^+$, which is adapted to also perform dense estimation and multi-modal learning, we also achieve top performance results for multi-modal depth estimation. But most importantly, our solutions are able to work in all cases with minimal latency both on GPU and CPU, facilitating its use on low-consumption hardware and real-time applications.

Based on the presented results, we believe that patch-based representations and transformers are a promising line of research for efficient event data processing. This framework also offers promising benefits to other event-based perception tasks, such as body tracking or object detection, and to different kinds of sparse data, like LiDAR data.

Chapter 6

Conclusions

Visual scene understanding is a very broad and complex problem that has applications in many fields such as autonomous driving, medical data processing, and augmented or virtual reality. These visual scene understanding applications encompasses many tasks, including object detection, action recognition, and depth estimation, which frequently involve or can be enhanced with the processing of video data rather than still images. In this thesis, we have seen how, although existing deep learning techniques achieve state-of-the-art performance in many of these tasks, their applicability in real-world scenarios is still limited due to various reasons. These deep learning solutions, and in particular the ones designed for video data processing, imply the construction of very complex models that demand large computational resources and often incur high latencies. Moreover, these models require large datasets and long training procedures to learn, and still, they are often prone to suffer from generalization problems when evaluated on data domains different from the training one. As a consequence of these issues, complex deep learning models also imply a high energy cost for their learning and deployment, which has become a growing concern as they are increasingly being integrated into our daily lives.

The present thesis addresses some of these challenges in designing deep learning models for video data processing that make a more responsible use of computational resources. Our proposed solutions include models that run efficiently, learn with minimal data and generalize to variable data domains when required, and still achieve high performance when compared to more complex solutions. We specifically focus on two scene understanding tasks that are essential for many applications, video object detection and action recognition. Moreover, we study how the use of non-traditional sensors can help in achieving deep learning efficiency. In particular, we work with event cameras due to their ability to capture only the dynamic information of the scene and their robustness to certain video conditions. In the following, we summarize our contributions to each one of the scene understanding topics addressed.

Object Detection

The first part of this thesis (Chapter 2) studies the state of object detection with video data. As described, state-of-the-art solutions [47, 53] achieve high performance on still images by sacrificing efficiency, and more efficient solutions [123, 95] do not achieve the same performance. This problem is increased when processing video data, where more complex solutions [171, 190], focused on analyzing a broader temporal context, are designed to overcome challenging video conditions.

Instead of building intricate solutions to better process video data, we propose a post-processing method [131] to analyze the relationship of the predicted bounding boxes. By doing so, we are then able to link object detections across frames and use their global information to refine their bounding box coordinates and class scores at each frame. These refined detections better fit the recorded objects. The performance of this proposed post-processing method is demonstrated with the predictions of two kinds of object detectors, an efficient detector that works on still frames [124] and a few complex and effective ones that benefit from the video data temporality [171, 190]. In both cases, we are able to improve the detection performance of the base object detectors, and we also outperform other post-processing methods [52, 11] of similar kind. But, most importantly, the use of our post-processing method supposes a minimal computational overhead. As a result, we are able to increase the efficacy of efficient object detectors to make them closer to more complex solutions, and still, when computational resources are not a limitation, these more complex video detection solutions can also benefit from this post-processing.

Action Recognition

The second part of this thesis (Chapters 3 and 4) studies the state of action recognition. As described, different from using raw RGB frames, using skeleton coordinates to represent human poses allows a more efficient processing since it presents a much lower dimensionality. However, when it comes to the applicability of action recognition to real use cases, we find several limitations. Commonly, we find applications where new action categories that must be detected are created on the fly, as is the case of rehabilitation or AR/VR, so we cannot build specific models to learn them. Additionally, available datasets to learn from are usually restricted to specific camera recording perspectives, so the generalization to other ones is limited, and the creation of more generalist datasets demands a lot of human work.

In order to overcome these issues, this thesis presents two novel action recognition solutions able to generalize and handle variable data domains, heterogeneous recording viewpoints, scales, translation directions, and motion artifacts. These two action recognition methods are specifically designed for two different scenarios. The first scenario involves the recognition of **full-body actions** [134] and, in particular, is evaluated in scenarios of therapies with autistic people, whose motion presents high variability due to specific and challenging artifacts. The second scenario involves the recognition of **hand actions** [130] and, in particular, the learning from very constrained recording perspectives and action domains, for a later generalization to a different recording viewpoint (e.g., egocentric view vs. third-person view) and action domain (e.g., sign language gestures vs. object interaction actions). In both cases, the proposed models are able to overcome these generalization challenges, achieving high effectiveness while also being highly efficient. This enables their use in real applications where computational resources are a limitation.

Scene understanding with event cameras

The last part of the present thesis (Chapter 5) studies how we can benefit from non-RGB sensors to improve the efficiency and efficacy of visual scene understanding. In particular, we focus on the processing of data from event cameras for several reasons. These sensors capture only sparse changes in the scene and ignore static and redundant information, making their processing able to ignore non-relevant parts of the scene to improve processing efficiency. Logged information is also independent of textures, which helps the later generalization; and presents high temporal resolution, which helps to capture fast motions in the scene. Moreover, these sensors are able to work in very challenging conditions. Although these characteristics promise potential benefits in many applications, prior works do not efficiently benefit from them.

This thesis presents a novel way to process this event information [133, 132] that effectively benefits from these properties, particularly the event data sparsity, to achieve high efficiency while maintaining high performance. For this purpose, we propose a new way of pre-processing event data that is highly descriptive but also presents certain sparsity; and a new attention-based backbone that benefits from this sparsity to improve its efficiency while maintaining high efficacy. This new processing methodology is evaluated in two tasks, action recognition and dense depth estimation. In both cases, we achieve state-of-the-art results both in performance and efficiency, probing the suitability of this novel processing method for event data. Moreover, we also probe that our event-based solution is easily adaptable to effectively perform multi-modal learning, which we probe by jointly processing both events and grayscale images.

6.1 Limitations and future work

Deep learning has set a new paradigm for visual scene recognition and, as we have seen in this thesis, there is still room for improvement in its application to real-world scenarios. Although our proposed solutions for different tasks advance video-based deep learning methods towards more efficient solutions, we still identify some limitations, but also different ways to further extend our work.

For instance, our **video object detection post-processing method** from Chapter 2 works by linking object detections by analyzing their location and geometry, appearance, and semantic information. However, the main limitation is that Neural Networks tend to be overconfident [50], so this process can lead to unexpected results. To improve the reliability of our method, we propose calculating and adding uncertainty information [69, 79, 141] of the object detection predictions. This enhancement will ensure that the linking process trusts only the most reliable information. Additionally, our proposed solution currently works offline, as it requires object detections to be calculated both for past and future video frames. But, similar to [11], our approach can be easily adapted for online settings, just by refining object detections with only past information and not future one. Furthermore, the idea of linking per-frame Neural Network results for its refinement with minimal computational cost can be extended to other tasks. For instance, a similar method can be built to improve semantic or panoptic segmentations by refining both their shape and class information.

In the case of our **action recognition** models from Chapters 3 and 4, the main limitation is the scarcity of labeled and diverse datasets, which constrains the amount of available data to learn from. However, there is plenty of motion information that could be extracted from sources such as YouTube, Netflix, and other video platforms. Although this information would not be hand-labeled, it can serve as a solid base for Neural Network models to learn with self-supervised techniques [23, 142, 41], along with the skeleton motion representations and data augmentation we have proposed. Developing such foundation models would ease a later fine-tuning for downstream tasks and enhance the N-shot evaluation performance. Moreover, incorporating narrations to video data, as in the case of the EPIC-KITCHENS Dataset [31], could inspire new CLIP-like methods [118] for weakly-supervised learning, ultimately enabling zero-shot language-based action recognition. While some studies [104, 88, 42] have already explored some of these ideas, they have primarily relied on RGB information rather than human skeleton coordinates which, as seen in previous Chapters, limits the generalization to different action and recording domains, and lead to more complex and computationally expensive models.

Regarding our proposed methods for **event data processing** (Chapter 5), the main limitation we face is the scarcity of available data for training. Unlike RGB cameras, these sensors are not as common and accessible, making it difficult to find open data to learn from. While some works propose simulation environments [109, 59, 44] to automatically generate event data or record RGB data displayed on a monitor [112, 143, 85, 15, 58] with an event camera, they often struggle to mimic the unique properties of event data, such as sparsity and time resolution. Despite these challenges, our event data processing framework has demonstrated its efficacy and efficiency in a few visual scene understanding tasks, and we believe it has the potential to be extended to other visual tasks like object detection and semantic segmentation. Moreover, our proposed patch-based event representation can be adapted to represent other sparse data sources, such as LiDAR recordings or point clouds, and then benefit as well from our sparse attention-based processing. Additionally, our proposed idea of leveraging latent vectors to serve as memory can also help in the processing of other temporal data sources in applications that demands online and real-time processing, such as RGB video data or audio. These latent vectors would replace more complex alternatives like Recurrent Neural Networks or attention models with expensive computational costs.

Chapter 7

Conclusiones

La comprensión visual de escenas es un problema muy amplio y complejo que tiene aplicaciones en muchos campos, como la conducción autónoma, el procesamiento de datos médicos y la realidad aumentada o virtual. Estas aplicaciones de comprensión visual de escenas abarcan muchas tareas, como la detección de objetos, el reconocimiento de acciones y la estimación de la profundidad, que con frecuencia implican o pueden mejorarse con el procesamiento de datos de vídeo en lugar de imágenes fijas. En esta tesis, hemos visto cómo, a pesar de que las técnicas de aprendizaje profundo existentes alcanzan un rendimiento puntero en muchas de estas tareas, su aplicabilidad en escenarios del mundo real sigue siendo limitada debido a diversas razones. Estas soluciones de aprendizaje profundo, y en particular las diseñadas para el procesamiento de datos de vídeo, implican la construcción de modelos muy complejos que demandan grandes recursos computacionales y a menudo incurren en altas latencias. Además, estos modelos requieren grandes conjuntos de datos y largos procesos de entrenamiento para aprender, y aún así, a menudo son propensos a sufrir problemas de generalización cuando se evalúan en dominios de datos diferentes al de entrenamiento. Como consecuencia de estos problemas, los modelos complejos de aprendizaje profundo también implican un alto coste energético para su aprendizaje y despliegue, lo que se ha convertido en una preocupación creciente a medida que se integran cada vez más en nuestra vida cotidiana.

Esta tesis aborda algunos de estos retos en el diseño de modelos de aprendizaje profundo para el procesamiento de datos de vídeo que hagan un uso más responsable de los recursos computacionales. Nuestras soluciones propuestas incluyen modelos que se ejecutan de manera eficiente, aprenden con datos mínimos y generalizan a dominios de datos variables cuando es necesario, y aún así logran un alto rendimiento en comparación con soluciones más complejas. Nos centramos específicamente en dos tareas de comprensión de escenas que son esenciales para muchas aplicaciones, la detección de objetos en vídeo y el reconocimiento de acciones. Además, estudiamos cómo el uso de sensores no tradicionales puede ayudar a lograr la eficiencia del aprendizaje profundo. En particular, trabajamos con cámaras de eventos debido a su capacidad para capturar únicamente la información dinámica de la escena y su robustez ante determinadas condiciones de vídeo. A continuación, resumimos nuestras contribuciones en cada uno de los temas de comprensión de escenas abordados.

Detección de objetos

La primera parte de esta tesis (Capítulo 2) estudia el estado de la detección de objetos con datos de vídeo. Como se ha descrito anteriormente, las soluciones del estado del arte [47, 53] consiguen un alto rendimiento en imágenes fijas sacrificando eficiencia, y las soluciones más eficientes [123, 95] no consiguen el mismo rendimiento. Este problema aumenta al procesar datos de vídeo, donde se diseñan soluciones más complejas [171, 190], centradas en analizar un contexto temporal más amplio, para superar las artefactos presentes en vídeo.

En lugar de construir soluciones complejas para procesar mejor los datos de vídeo, proponemos un método de post-procesamiento [131] para analizar las detecciones predichas. De este modo, podemos vincular las detecciones de objetos entre fotogramas y utilizar su información global para refinar las coordenadas de sus cuadros delimitadores y las probabilidades por clase en cada fotograma. Estas detecciones refinadas se ajustan mejor a los objetos reales. El rendimiento de este método de post-procesamiento propuesto se evalua con las predicciones de dos tipos de detectores de objetos, un detector eficiente que funciona en imágenes fijas [124] y dos complejos y eficaces que se benefician de la temporalidad de los datos de vídeo [171, 190]. En ambos casos, somos capaces de mejorar el rendimiento de detección de los detectores de objetos base, y también superamos a otros métodos de post-procesamiento [52, 11] similares. Pero lo que es más importante, el uso de nuestro método de post-procesamiento supone una sobrecarga computacional mínima. Como resultado, somos capaces de aumentar la eficacia de los detectores de objetos eficientes para acercarlos a soluciones más complejas, y aún así, cuando los recursos computacionales no son una limitación, estas soluciones de detección de vídeo más complejas también pueden beneficiarse de este post-procesamiento.

Reconocimiento de Acciones

La segunda parte de esta tesis (Capítulos 3 y 4) estudia el estado del reconocimiento de acciones. Como se ha descrito anteriormente, a diferencia del procesamiento de imágenes RGB, el uso de coordenadas de esqueletos para representar poses humanas permite un procesamiento más eficiente ya que presenta una dimensionalidad mucho menor. Sin embargo, cuando se trata de la aplicabilidad del reconocimiento de acciones a casos de uso reales, encontramos varias limitaciones. Frecuentemente, nos encontramos con aplicaciones en las que las nuevas categorías de acciones que deben ser detectadas se crean sobre la marcha, como es el caso de la rehabilitación o la RA/RV, por lo que no podemos construir modelos específicos para aprenderlas. Además, los conjuntos de datos disponibles para aprender suelen estar restringidos a perspectivas de grabación de cámara específicas, por lo que la generalización a otras es limitada, y la creación de conjuntos de datos más generalistas exige mucho trabajo humano.

Para superar estos problemas, esta tesis presenta dos nuevas soluciones de reconocimiento de acciones capaces de generalizar y manejar dominios de datos variables, puntos de vista de grabación heterogéneos, escalas, direcciones de traslación y artefactos de movimiento. Estos dos métodos de reconocimiento de acciones se han diseñado específicamente para dos escenarios diferentes. El primero consiste en el reconocimiento de acciones de cuerpo entero. [134] y, en particular, se evalúa en escenarios de terapias con personas autistas, cuyo movimiento presenta una alta variabilidad debido a artefactos específicos. El segundo escenario implica el reconocimiento de **acciones de la mano** [130] y, en particular, el aprendizaje a partir de perspectivas de grabación y dominios de acción muy restringidos, para una posterior generalización a un punto de vista de grabación diferente (por ejemplo, vista en primera persona frente a vista en tercera persona) y dominio de acción (por ejemplo, gestos de lengua de signos frente a acciones de interacción con objetos). En ambos casos, los modelos propuestos son capaces de superar estos retos de generalización, logrando una alta efectividad a la vez que son altamente eficientes. Esto permite su uso en aplicaciones reales donde los recursos computacionales son una limitación.

Comprensión de escenas con cámaras de eventos

La última parte de esta tesis (Capítulo 5) estudia cómo podemos beneficiarnos de los sensores no RGB para mejorar la eficiencia y eficacia en la comprensión visual de escenas. En concreto, nos centramos en el procesamiento de datos procedentes de cámaras de eventos por varias razones. Estos sensores captan sólo cambios dispersos en la escena e ignoran la información estática y redundante, lo que hace que su procesamiento pueda ignorar partes no relevantes de la escena para mejorar su eficiencia. La información registrada también es independiente de las texturas, lo que ayuda a la posterior generalización; y presenta una alta resolución temporal, lo que ayuda a capturar movimientos rápidos en la escena. Además, estos sensores son capaces de trabajar en condiciones muy difíciles. Aunque estas características prometen beneficios potenciales en muchas aplicaciones, los trabajos anteriores no las aprovechan eficientemente.

Esta tesis presenta una novedosa forma de procesar esta información de eventos [133, 132] que se beneficia eficazmente de estas propiedades, en particular de la dispersión de datos de eventos, para lograr una alta eficiencia manteniendo un alto rendimiento. Para ello, proponemos una nueva forma de pre-procesar los datos de eventos que es altamente descriptiva pero que también presenta cierta dispersión; y un nuevo modelo basado en la métodos de atención que se beneficia de esta dispersión para mejorar su eficiencia manteniendo una alta eficacia. Esta nueva metodología de procesamiento se evalúa en dos tareas, reconocimiento de acciones y estimación de profundidad. En ambos casos, se obtienen resultados punteros tanto en rendimiento como en eficiencia, lo que demuestra la idoneidad de este nuevo método de procesamiento para los datos de eventos. Además, también comprobamos que nuestra solución basada en eventos es fácilmente adaptable para llevar a cabo un aprendizaje multimodal eficaz, que comprobamos procesando conjuntamente eventos e imágenes en escala de grises.

7.1 Limitaciones y trabajo futuro

El aprendizaje profundo ha establecido un nuevo paradigma para el reconocimiento visual de escenas y, como hemos visto en esta tesis, todavía hay margen de mejora en su aplicación a escenarios del mundo real. Aunque nuestras soluciones propuestas para diferentes tareas hacen avanzar los métodos de aprendizaje profundo basados en vídeo hacia soluciones más eficientes, seguimos identificando algunas limitaciones, pero también diferentes formas de seguir ampliando nuestro trabajo.

Por ejemplo, nuestro **método de post-procesamiento de detección de objetos en vídeo** del Capítulo 2 funciona enlazando detecciones de objetos mediante el análisis de su ubicación y geometría, apariencia e información semántica. Sin embargo, la principal limitación es que las Redes Neuronales tienden a ser demasiado confiadas [50], por lo que este proceso puede conducir a resultados inesperados. Para mejorar la fiabilidad de nuestro método, proponemos calcular y añadir información de incertidumbre [69, 79, 141] de las predicciones de detección de objetos. Esta mejora garantizará que el proceso de vinculación confíe únicamente en la información más fiable. Además, nuestra solución propuesta funciona actualmente *offline*, ya que requiere que las detecciones de objetos se calculen tanto para los fotogramas de vídeo pasados como para los futuros. Pero, al igual que ocurre con [11], nuestro enfoque puede adaptarse fácilmente a entornos *online*, simplemente refinando las detecciones de objetos con información pasada y no futura. Además, la idea de enlazar los resultados de la red neuronal por fotograma para su refinamiento con un coste computacional mínimo puede extenderse a otras tareas. Por ejemplo, se puede construir un método similar para mejorar las segmentaciones semánticas o panópticas refinando tanto su información de forma como de clase.

En el caso de nuestros modelos de **reconocimiento de acciones** de los Capítulos 3 y 4, la principal limitación es la escasez de conjuntos de datos etiquetados y diversos, lo que limita la cantidad de datos disponibles de los que aprender. Sin embargo, hay mucha información de movimiento que podría extraerse de fuentes como YouTube, Netflix y otras plataformas de vídeo. Aunque esta información no estaría etiquetada a mano, puede servir de base sólida para que los modelos de redes neuronales aprendan con técnicas autosupervisadas [23, 142, 41], junto con las representaciones de movimiento de esqueleto y el aumento de datos que hemos propuesto. El desarrollo de estos modelos facilitaría un ajuste posterior para otras tareas y mejoraría el rendimiento de la evaluación de N-shot. Además, la incorporación de narraciones a los datos de vídeo, como en el caso del conjunto de datos EPIC-KITCHENS [31], podría inspirar nuevos métodos similares a CLIP [118] para el aprendizaje débilmente supervisado, permitiendo en última instancia el reconocimiento de acciones basado en el lenguaje en cero tomas. Aunque algunos estudios [104, 88, 42] ya han explorado algunas de estas ideas, se han basado principalmente en la información RGB en lugar de en las coordenadas del esqueleto humano que, como se ha visto en Capítulos anteriores, limita la generalización a diferentes dominios de acción y grabación, y conduce a modelos más complejos y costosos computacionalmente.

En lo que respecta a nuestros métodos propuestos para el procesamiento de datos de eventos (Capítulo 5), la principal limitación a la que nos enfrentamos es la escasez de datos disponibles para el entrenamiento. A diferencia de las cámaras RGB, estos sensores no son tan comunes y accesibles, por lo que resulta difícil encontrar datos abiertos con los que aprender. Mientras que algunos trabajos proponen entornos de simulación [109, 59, 44] para generar automáticamente datos de eventos o grabar datos RGB mostrados en un monitor [112, 143, 85, 15, 58] con una cámara de eventos, a menudo fallan al imitar las propiedades únicas de los datos de eventos, como la dispersión y la resolución temporal. A pesar de estos retos, nuestros modelos de procesamiento de datos de eventos ha demostrado su eficacia y eficiencia en algunas tareas de comprensión de escenas visuales, y creemos que tiene potencial para extenderse a otras tareas visuales como la detección de objetos y la segmentación semántica. Además, nuestra propuesta de representación de eventos basada en parches puede adaptarse para representar otras fuentes de datos dispersos, como grabaciones LiDAR o nubes de puntos, y beneficiarse también de nuestro procesamiento basado en la atención dispersa. Además, nuestra idea propuesta de aprovechar los vectores latentes para que sirvan de memoria también puede ayudar en el procesamiento de otras fuentes de datos temporales en aplicaciones que exigen un procesamiento *online* y en tiempo real, como los datos de vídeo RGB o audio. Estos vectores latentes sustituirían a alternativas más complejas, como las redes neuronales recurrentes o los modelos de atención con costes computacionales elevados.

Bibliography

[1] Bahareh Abbasi, Natawut Monaikul, Zhanibek Rysbek, Barbara Di Eugenio, and Milos Zefran. A multimodal human-robot interaction manager for assistive robots. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 6756–6762, 2019.

[2] Hassan Akbari, Liangzhe Yuan, Rui Qian, Wei-Hong Chuang, Shih-Fu Chang, Yin Cui, and Boqing Gong. Vatt: Transformers for multimodal self-supervised learning from raw video, audio and text. Advances in Neural Information Processing Systems, 34:24206–24221, 2021.

[3] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. Advances in Neural Information Processing Systems, 35:23716–23736, 2022.

[4] Iñigo Alonso, Alberto Sabater, David Ferstl, Luis Montesano, and Ana C. Murillo. Semi-supervised semantic segmentation with pixel-level contrastive learning from a class-wise memory bank. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 8219–8228, 2021.

[5] Arnon Amir et al. A low power, fully event-based gesture recognition system. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 7243–7252, 2017.

[6] Anastasios N Angelopoulos, Julien NP Martel, Amit P Kohli, Jorg Conradt, and Gordon Wetzstein. Event-based near-eye gaze tracking beyond 10,000 hz. IEEE Transactions on Visualization & Computer Graphics, 27(05):2577–2586, 2021.

[7] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 6836–6846, 2021.

[8] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv e-prints, pages arXiv–1803, 2018.

[9] Raymond Baldwin, Ruixu Liu, Mohammed Mutlaq Almatrafi, Vijayan K Asari, and Keigo Hirakawa. Time-ordered recent event (TORE) volumes for event cameras. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(2):2519–2532, 2022.

[10] Tamas Bates, Karinne Ramirez-Amaro, Tetsunari Inamura, and Gordon Cheng. On-line simultaneous learning and recognition of everyday activities from virtual re-

ality performances. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3510–3515, 2017.

[11] Hatem Belhassen, Heng Zhang, Fresse Virginie, and El-Bey Bourennane. Improving video object detection by seq-bbox matching. In VISIGRAPP (5: VISAPP), pages 226–233, 2019.

[12] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2):157–166, 1994.

[13] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In IEEE International Conference on Image Processing, pages 3464–3468, 2016.

[14] Yin Bi, Aaron Chadha, Alhabib Abbas, Eirina Bourtsoulatze, and Yiannis Andreopoulos. Graph-based object classification for neuromorphic vision sensing. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 491–501, 2019.

[15] Yin Bi, Aaron Chadha, Alhabib Abbas, Eirina Bourtsoulatze, and Yiannis Andreopoulos. Graph-based spatio-temporal feature learning for neuromorphic vision sensing. IEEE Transactions on Image Processing, 29:9084–9098, 2020.

[16] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Audio chord recognition with recurrent neural networks. In ISMIR, pages 335–340. Curitiba, 2013.

[17] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. Advances in Neural Information Processing Systems, 33:1877–1901, 2020.

[18] Enrico Calabrese, Gemma Taverni, Christopher Awai Easthope, Sophie Skriabine, Federico Corradi, Luca Longinotti, Kynan Eng, and Tobi Delbruck. Dhp19: Dynamic vision sensor 3d human pose dataset. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pages 0–0, 2019.

[19] Marco Cannici, Marco Ciccone, Andrea Romanoni, and Matteo Matteucci. A differentiable recurrent surface for asynchronous event-based data. In Proceedings of the European Conference on Computer Vision, pages 136–152. Springer, 2020.

[20] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(1):172–186, 2021.

[21] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 7291–7299, 2017.

[22] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Proceedings of the European Conference on Computer Vision, pages 213–229. Springer, 2020.

[23] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 9650–9660, 2021.

[24] Tatjana Chavdarova, Pierre Baqué, Stéphane Bouquet, Andrii Maksai, Cijo Jose, Timur Bagautdinov, Louis Lettry, Pascal Fua, Luc Van Gool, and François Fleuret. WILDTRACK: A multi-camera hd dataset for dense unscripted pedestrian detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5030–5039, 2018.

[25] Cheng Chen, Yueting Zhuang, Feiping Nie, Yi Yang, Fei Wu, and Jun Xiao. Learning a 3d human pose distance metric from geometric pose descriptor. IEEE Transactions on Visualization and Computer Graphics, 17(11):1676–1689, 2010.

[26] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Proceedings of the International Conference on Machine Learning, pages 1597–1607. PMLR, 2020.

[27] Xinghao Chen, Guijin Wang, Hengkai Guo, Cairong Zhang, Hang Wang, and Li Zhang. Mfa-net: Motion feature augmented network for dynamic hand gesture recognition from skeletal data. Sensors, 19(2):239, 2019.

[28] Ke Cheng, Yifan Zhang, Xiangyu He, Weihan Chen, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with shift graph convolutional network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 183–192, 2020.

[29] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078, 2014.

[30] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. Advances in Neural Information Processing Systems, 29, 2016.

[31] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Jian Ma, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Rescaling egocentric vision: Collection, pipeline and challenges for EPIC-KITCHENS-100. International Journal of Computer Vision, pages 1–23, 2022.

[32] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In Proceedings of the European Conference on Computer Vision, pages 720–736, 2018.

[33] Quentin De Smedt, Hazem Wannous, Jean-Philippe Vandeborre, Joris Guerry, Bertrand Le Saux, and David Filliat. Shrec'17 track: 3d hand gesture recognition using a depth and skeletal dataset. In 3DOR-10th Eurographics Workshop on 3D Object Retrieval, pages 1–6, 2017.

[34] Yongjian Deng, Hao Chen, Huiying Chen, and Youfu Li. EV-VGCNN: A voxel graph cnn for event-based object classification. arXiv preprint arXiv:2106.00216, 1(2):6, 2021.

[35] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2625–2634, 2015.

[36] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In International Conference on Learning Representations, 2020.

[37] Sean Ryan Fanello, Ilaria Gori, Giorgio Metta, and Francesca Odone. One-shot learning for real-time action recognition. In Pattern Recognition and Image Analysis: 6th Iberian Conference, pages 31–40. Springer, 2013.

[38] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 178–178, 2004.

[39] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1933–1941, 2016.

[40] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. In Proceedings of the IEEE International Conference on Computer Vision, pages 3038–3046, 2017.

[41] Basura Fernando, Hakan Bilen, Efstratios Gavves, and Stephen Gould. Self-supervised video representation learning with odd-one-out networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3636–3645, 2017.

[42] Junyu Gao, Tianzhu Zhang, and Changsheng Xu. I know the relationships: Zero-shot action recognition via two-stream graph convolutional networks and knowledge graphs. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 8303–8311, 2019.

[43] Guillermo Garcia-Hernando, Shanxin Yuan, Seungryul Baek, and Tae-Kyun Kim. First-person hand action benchmark with rgb-d videos and 3d hand pose annotations. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 409–419, 2018.

[44] Daniel Gehrig, Mathias Gehrig, Javier Hidalgo-Carrió, and Davide Scaramuzza. Video to events: Recycling video datasets for event cameras. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3586–3595, 2020.

[45] Daniel Gehrig, Michelle Rüegg, Mathias Gehrig, Javier Hidalgo-Carrió, and Davide Scaramuzza. Combining events and frames using recurrent asynchronous multimodal

networks for monocular depth prediction. IEEE Robotics and Automation Letters, 6(2):2822–2829, 2021.

[46] Rohan Ghosh, Anupam Gupta, Andrei Nakagawa, Alcimar Soares, and Nitish Thakor. Spatiotemporal filtering for event-based action recognition. arXiv preprint arXiv:1903.07067, 2019.

[47] Ross Girshick. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, pages 1440–1448, 2015.

[48] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 580–587, 2014.

[49] Ian Goodfellow, Yoshua Bengio, Aaron Courville, et al. Deep learning book. MIT Press, 521(7553):800, 2016.

[50] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In Proceedings of the International Conference on Machine Learning, pages 1321–1330. PMLR, 2017.

[51] Meera Hahn, Andrew Silva, and James M Rehg. Action2vec: A crossmodal embedding approach to action learning. arXiv e-prints, pages arXiv–1901, 2019.

[52] Wei Han, Pooya Khorrami, Tom Le Paine, Prajit Ramachandran, Mohammad Babaeizadeh, Honghui Shi, Jianan Li, Shuicheng Yan, and Thomas S Huang. Seq-nms for video object detection. arXiv preprint arXiv:1602.08465, 2016.

[53] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, pages 2961–2969, 2017.

[54] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016.

[55] Javier Hidalgo-Carrió, Daniel Gehrig, and Davide Scaramuzza. Learning monocular dense depth from events. In International Conference on 3D Vision, pages 534–542. IEEE, 2020.

[56] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[57] Jingxuan Hou, Guijin Wang, Xinghao Chen, Jing-Hao Xue, Rui Zhu, and Huazhong Yang. Spatial-temporal attention res-TCN for skeleton-based dynamic hand gesture recognition. In Proceedings of the European Conference on Computer Vision Workshops, pages 0–0, 2018.

[58] Yuhuang Hu, Hongjie Liu, Michael Pfeiffer, and Tobi Delbruck. Dvs benchmark datasets for object tracking, action recognition, and object recognition. Frontiers in neuroscience, 10:405, 2016.

[59] Yuhuang Hu, Shih-Chii Liu, and Tobi Delbruck. v2e: From video frames to realistic dvs events. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1312–1321, 2021.

[60] Jian Huang, Jianhua Tao, Bin Liu, Zheng Lian, and Mingyue Niu. Multimodal transformer fusion for continuous emotion recognition. In IEEE International Conference on Acoustics, Speech and Signal Processing, pages 3507–3511, 2020.

[61] Simone Undri Innocenti, Federico Becattini, Federico Pernici, and Alberto Del Bimbo. Temporal binary representation for event-based action recognition. In International Conference on Pattern Recognition, pages 10426–10432. IEEE, 2021.

[62] P Izmailov, AG Wilson, D Podoprikhin, D Vetrov, and T Garipov. Averaging weights leads to wider optima and better generalization. In Conference on Uncertainty in Artificial Intelligence, pages 876–885, 2018.

[63] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. arXiv preprint arXiv:2107.14795, 2021.

[64] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In Proceedings of the International Conference on Machine Learning, pages 4651–4664. PMLR, 2021.

[65] Bhavan Jasani and Afshaan Mazagonwalla. Skeleton based zero shot action recognition in joint pose-language semantic space. arXiv e-prints, pages arXiv–1911, 2019.

[66] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(1):221–231, 2012.

[67] Jacques Kaiser, Hesham Mostafa, and Emre Neftci. Synaptic plasticity dynamics for deep continuous local learning (DECOLLE). Frontiers in Neuroscience, 14:424, 2020.

[68] Kai Kang, Hongsheng Li, Junjie Yan, Xingyu Zeng, Bin Yang, Tong Xiao, Cong Zhang, Zhe Wang, Ruohui Wang, Xiaogang Wang, et al. T-CNN: Tubelets with convolutional neural networks for object detection from videos. IEEE Transactions on Circuits and Systems for Video Technology, 28(10):2896–2907, 2017.

[69] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? Advances in Neural Information Processing Systems, 30, 2017.

[70] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of NAACL-HLT, pages 4171–4186, 2019.

[71] Tae Soo Kim and Austin Reiter. Interpretable 3d human action analysis with temporal convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 1623–1631, 2017.

[72] Simon Klenk, Jason Chui, Nikolaus Demmel, and Daniel Cremers. TUM-VIE: The TUM stereo visual-inertial event dataset. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 8601–8608, 2021.

[73] Jakub Konečnỳ and Michal Hagara. One-shot-learning gesture recognition using hog-hof features. The Journal of Machine Learning Research, 15(1):2513–2532, 2014.

[74] Kaushik Koneripalli, Suhas Lohit, Rushil Anirudh, and Pavan Turaga. Rate-invariant autoencoding of time-series. In IEEE International Conference on Acoustics, Speech and Signal Processing, pages 3732–3736, 2020.

[75] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6):84–90, 2017.

[76] Dennis Krupke, Frank Steinicke, Paul Lubos, Yannick Jonetzko, Michael Görner, and Jianwei Zhang. Comparison of multimodal heading and pointing gestures for co-located mixed reality human-robot interaction. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1–9, 2018.

[77] Xavier Lagorce, Garrick Orchard, Francesco Galluppi, Bertram E Shi, and Ryad B Benosman. Hots: a hierarchy of event-based time-surfaces for pattern recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(7):1346–1359, 2016.

[78] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 29, 2015.

[79] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. Advances in Neural Information Processing Systems, 30, 2017.

[80] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In Proceedings of the European Conference on Computer Vision, pages 734–750, 2018.

[81] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. Nature, 521(7553):436–444, 2015.

[82] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. Advances in Neural Information Processing Systems, 2, 1989.

[83] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998.

[84] Chuankun Li, Shuai Li, Yanbo Gao, Xiang Zhang, and Wanqing Li. A two-stream neural network for pose-based hand gesture recognition. IEEE Transactions on Cognitive and Developmental Systems, 14(4):1594–1603, 2021.

[85] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: an event-stream dataset for object classification. Frontiers in neuroscience, 11:309, 2017.

[86] Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, and Ming Liu. Neural speech synthesis with transformer network. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 6706–6713, 2019.

[87] Suichan Li and Feng Chen. 3d-detnet: a single stage video-based vehicle detector. In Third International Workshop on Pattern Recognition, volume 10828, pages 60–66. SPIE, 2018.

[88] Chung-Ching Lin, Kevin Lin, Lijuan Wang, Zicheng Liu, and Linjie Li. Cross-modal representation learning for zero-shot action recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 19978–19988, 2022.

[89] Ji Lin, Chuang Gan, and Song Han. TSM: Temporal shift module for efficient video understanding. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 7083–7093, 2019.

[90] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In Proceedings of the European Conference on Computer Vision, pages 740–755. Springer, 2014.

[91] Jun Liu, Amir Shahroudy, Mauricio Perez, Gang Wang, Ling-Yu Duan, and Alex C Kot. Ntu rgb+ d 120: A large-scale benchmark for 3d human activity understanding. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(10):2684–2701, 2019.

[92] Jun Liu, Amir Shahroudy, Dong Xu, Alex C Kot, and Gang Wang. Skeleton-based action recognition using spatio-temporal lstm network with trust gates. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(12):3007–3021, 2017.

[93] Jun Liu, Gang Wang, Ping Hu, Ling-Yu Duan, and Alex C Kot. Global context-aware attention lstm networks for 3d action recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1647–1656, 2017.

[94] Jianbo Liu, Ying Wang, Yongcheng Liu, Shiming Xiang, and Chunhong Pan. 3d posturenet: A unified framework for skeleton-based posture recognition. Pattern Recognition Letters, 140:143–149, 2020.

[95] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, pages 21–37. Springer, 2016.

[96] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 10012–10022, 2021.

[97] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3202–3211, 2022.

[98] Ziyu Liu, Hongwen Zhang, Zhenghao Chen, Zhiyong Wang, and Wanli Ouyang. Disentangling and unifying graph convolutions for skeleton-based action recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 143–152, 2020.

[99] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101, 2017.

[100] Yongyi Lu, Cewu Lu, and Chi-Keung Tang. Online video object detection using association lstm. In Proceedings of the IEEE International Conference on Computer Vision, pages 2344–2352, 2017.

[101] Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. Estimating the carbon footprint of bloom, a 176b parameter language model. arXiv preprint arXiv:2211.02001, 2022.

[102] Chunyong Ma, Shengsheng Zhang, Anni Wang, Yongyang Qi, and Ge Chen. Skeleton-based dynamic hand gesture recognition using an enhanced network with one-shot learning. Applied Sciences, 10(11):3680, 2020.

[103] Behrooz Mahasseni, Michael Lam, and Sinisa Todorovic. Unsupervised video summarization with adversarial lstm networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 202–211, 2017.

[104] Devraj Mandal, Sanath Narayan, Sai Kumar Dwivedi, Vikram Gupta, Shuaib Ahmed, Fahad Shahbaz Khan, and Ling Shao. Out-of-distribution detection for generalized zero-shot action recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9985–9993, 2019.

[105] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems, 26, 2013.

[106] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. MOT16: A benchmark for multi-object tracking. arXiv preprint arXiv:1603.00831, 2016.

[107] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5079–5088, 2018.

[108] Elias Mueggler, Christian Forster, Nathan Baumli, Guillermo Gallego, and Davide Scaramuzza. Lifetime estimation of events from dynamic vision sensors. In Proceedings of the IEEE International Conference on Robotics and Automation, pages 4874–4881, 2015.

[109] Jalees Nehvi, Vladislav Golyanik, Franziska Mueller, Hans-Peter Seidel, Mohamed Elgharib, and Christian Theobalt. Differentiable event stream simulator for non-rigid 3d tracking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1302–1311, 2021.

[110] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased lstm: Accelerating recurrent network training for long or event-based sequences. Advances in Neural Information Processing Systems, 29, 2016.

[111] Anh Nguyen, Thanh-Toan Do, Darwin G Caldwell, and Nikos G Tsagarakis. Real-time 6dof pose relocalization for event cameras with stacked spatial lstm networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pages 0–0, 2019.

[112] Garrick Orchard, Ajinkya Jayawant, Gregory K Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. Frontiers in neuroscience, 9:437, 2015.

[113] Omar Oreifej and Zicheng Liu. Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 716–723, 2013.

[114] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 528–540, 2018.

[115] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10):1345–1359, 2010.

[116] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In Proceedings of the International Conference on Machine Learning, pages 1310–1318. Pmlr, 2013.

[117] Mauricio Perez, Jun Liu, and Alex C Kot. Interaction relational network for mutual action recognition. IEEE Transactions on Multimedia, 24:366–376, 2021.

[118] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In Proceedings of the International Conference on Machine Learning, pages 8748–8763. PMLR, 2021.

[119] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. arXiv preprint arXiv:2212.04356, 2022.

[120] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research, 21(1):5485–5551, 2020.

[121] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 12179–12188, 2021.

[122] Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization. In Proceedings of the British Machine Vision Conference, pages 16–1, 2017.

[123] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 779–788, 2016.

[124] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767, 2018.

[125] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in Neural Information Processing Systems, 28, 2015.

[126] Juan Pablo Rodríguez-Gómez, Raul Tapia, Julio L Paneque, Pedro Grau, Augusto Gómez Eguíluz, Jose Ramiro Martínez-de Dios, and Anibal Ollero. The GRIFFIN perception dataset: Bridging the gap between flapping-wing flight and robotic perception. IEEE Robotics and Automation Letters, 6(2):1066–1073, 2021.

[127] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6):386, 1958.

[128] Viktor Rudnev, Vladislav Golyanik, Jiayi Wang, Hans-Peter Seidel, Franziska Mueller, Mohamed Elgharib, and Christian Theobalt. Eventhands: Real-time neural 3d hand pose estimation from an event stream. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 12385–12395, 2021.

[129] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. International journal of computer vision, 115:211–252, 2015.

[130] Alberto Sabater, Iñigo Alonso, Luis Montesano, and Ana C Murillo. Domain and view-point agnostic hand action recognition. IEEE Robotics and Automation Letters, 6(4):7823–7830, 2021.

[131] Alberto Sabater, Luis Montesano, and Ana C Murillo. Robust and efficient post-processing for video object detection. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 10536–10542, 2020.

[132] Alberto Sabater, Luis Montesano, and Ana C Murillo. Event transformer+. a multi-purpose solution for efficient event data processing. arXiv preprint arXiv:2211.12222, 2022.

[133] Alberto Sabater, Luis Montesano, and Ana C Murillo. Event transformer. a sparse-aware solution for efficient event data processing. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2677–2686, 2022.

[134] Alberto Sabater, Laura Santos, Jose Santos-Victor, Alexandre Bernardino, Luis Montesano, and Ana C Murillo. One-shot action recognition in challenging therapy scenarios. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2777–2785, 2021.

[135] Laura Santos, Silvia Annunziata, Alice Geminiani, Elena Brazzoli, Arianna Caglio, José Santos-Victor, Alessandra Pedrocchi, and Ivana Olivieri. Interactive social games with a social robot (IOGIOCO): Communicative gestures training for preschooler children with autism spectrum disorder. In I Congresso Annuale Rete IRCCCS Neuroscienze e Neuroriabilitazione, 2020.

[136] Laura Santos, Alice Geminiani, Ivana Olivieri, José Santos-Victor, and Alessandra Pedrocchi. Copyrobot: Interactive mirroring robotics game for asd children. In XV Mediterranean Conference on Medical and Biological Engineering and Computing–MEDICON 2019: Proceedings of MEDICON 2019, September 26-28, 2019, Coimbra, Portugal, pages 2014–2027. Springer, 2020.

[137] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. arXiv preprint arXiv:2211.05100, 2022.

[138] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 815–823, 2015.

[139] Christoph Schuhmann, Robert Kaczmarczyk, Aran Komatsuzaki, Aarush Katta, Richard Vencu, Romain Beaumont, Jenia Jitsev, Theo Coombes, and Clayton Mullis. LAION-400M: Open dataset of clip-filtered 400 million image-text pairs. In NeurIPS Workshop Datacentric AI, number FZJ-2022-00923. Jülich Supercomputing Center, 2021.

[140] Yusuke Sekikawa, Kosuke Hara, and Hideo Saito. Eventnet: Asynchronous recursive event processing. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3887–3896, 2019.

[141] Murat Sensoy, Lance Kaplan, and Melih Kandemir. Evidential deep learning to quantify classification uncertainty. Advances in Neural Information Processing Systems, 31, 2018.

[142] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In Proceedings of the IEEE International Conference on Robotics and Automation, pages 1134–1141, 2018.

[143] Teresa Serrano-Gotarredona and Bernabé Linares-Barranco. A 128×1281.5% contrast sensitivity 0.9% fpn 3 μs latency 4 mw asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers. IEEE Journal of Solid-State Circuits, 48(3):827–838, 2013.

[144] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. Advances in Neural Information Processing Systems, 28, 2015.

[145] Sumit B Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. Advances in neural information processing systems, 31, 2018.

[146] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In Artificial intelligence and machine learning for multi-domain operations applications, volume 11006, pages 369–386. SPIE, 2019.

[147] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. Advances in Neural Information Processing Systems, 30, 2017.

[148] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. Advances in Neural Information Processing Systems, 29, 2016.

[149] Hongmei Song, Wenguan Wang, Sanyuan Zhao, Jianbing Shen, and Kin-Man Lam. Pyramid dilated deeper convlstm for video salient object detection. In Proceedings of the European Conference on Computer Vision, pages 715–731, 2018.

[150] Kun Su, Xiulong Liu, and Eli Shlizerman. Predict & cluster: Unsupervised skeleton based action recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9631–9640, 2020.

[151] Xiao Sun, Yichen Wei, Shuang Liang, Xiaoou Tang, and Jian Sun. Cascaded hand pose regression. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 824–832, 2015.

[152] Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In Proceedings of the International Conference on Machine Learning, pages 1017–1024, 2011.

[153] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. Advances in Neural Information Processing Systems, 33:7537–7547, 2020.

[154] Ajay Kumar Tanwani and Sylvain Calinon. A generative model for intention recognition and manipulation assistance in teleoperation. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 43–50, 2017.

[155] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision, pages 4489–4497, 2015.

[156] Amin Ullah, Jamil Ahmad, Khan Muhammad, Muhammad Sajjad, and Sung Wook Baik. Action recognition in video sequences using deep bi-directional lstm with cnn features. IEEE Access, 6:1155–1166, 2017.

[157] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In 9th ISCA Speech Synthesis Workshop, 2020.

[158] Ajay Vasudevan, Pablo Negri, Camila Di Ielsi, Bernabe Linares-Barranco, and Teresa Serrano-Gotarredona. SL-Animals-DVS: event-driven sign language animals dataset. Pattern Analysis and Applications, pages 1–16, 2021.

[159] Ajay Vasudevan, Pablo Negri, Bernabe Linares-Barranco, and Teresa Serrano-Gotarredona. Introduction and analysis of an event-based sign language dataset. In IEEE International Conference on Automatic Face and Gesture Recognition, pages 675–682, 2020.

[160] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in Neural Information Processing Systems, 30, 2017.

[161] Sai Vemprala, Sami Mian, and Ashish Kapoor. Representation learning for event-based visuomotor policies. Advances in Neural Information Processing Systems, 34:4712–4724, 2021.

[162] Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Ultimate slam? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios. In Proceedings of the IEEE International Conference on Robotics and Automation, volume 3, pages 994–1001, 2018.

[163] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. Advances in Neural Information Processing Systems, 29, 2016.

[164] Lin Wang, Yujeong Chae, and Kuk-Jin Yoon. Dual transfer learning for event-based end-task prediction via pluggable event to image translation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 2135–2145, 2021.

[165] Qinyi Wang, Yexin Zhang, Junsong Yuan, and Yilong Lu. Space-time event clouds for gesture recognition: From rgb cameras to event cameras. In IEEE Winter Conference on Applications of Computer Vision, pages 1826–1835, 2019.

[166] Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. arXiv e-prints, pages arXiv–1911, 2019.

[167] Ziwei Wang, Liyuan Pan, Yonhon Ng, Zheyu Zhuang, and Robert Mahony. Stereo hybrid event-frame (shef) cameras for 3d perception. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 9758–9764, 2021.

[168] Wenming Weng, Yueyi Zhang, and Zhiwei Xiong. Event-based video reconstruction using transformer. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 2563–2572, 2021.

[169] Paul J Werbos. Backpropagation through time: what it does and how to do it. Proceedings of the IEEE, 78(10):1550–1560, 1990.

[170] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In IEEE International Conference on Image Processing, pages 3645–3649, 2017.

[171] Haiping Wu, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Sequence level semantics aggregation for video object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 9217–9225, 2019.

[172] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. Frontiers in neuroscience, 12:331, 2018.

[173] Fu Xiong, Boshen Zhang, Yang Xiao, Zhiguo Cao, Taidong Yu, Joey Tianyi Zhou, and Junsong Yuan. A2j: Anchor-to-joint regression network for 3d articulated pose estimation from a single depth image. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 793–802, 2019.

[174] Fan Yang, Yang Wu, Sakriani Sakti, and Satoshi Nakamura. Make skeleton-based action recognition model smaller, faster and better. In Proceedings of the ACM multimedia asia, pages 1–6, 2019.

[175] Chengxi Ye, Anton Mitrokhin, Cornelia Fermüller, James A Yorke, and Yiannis Aloimonos. Unsupervised learning of dense optical flow, depth and egomotion with event-based sensors. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 5831–5838, 2020.

[176] Heiga Zen, Yannis Agiomyrgiannakis, Niels Egberts, Fergus Henderson, and Przemysław Szczepaniak. Fast, compact, and high quality lstm-rnn based statistical parametric speech synthesizers for mobile devices. Interspeech 2016, 2016.

[177] Chen Zhang and Joohee Kim. Video object detection with two-path convolutional lstm pyramid. IEEE Access, 8:151681–151691, 2020.

[178] Chang-Bin Zhang, Peng-Tao Jiang, Qibin Hou, Yunchao Wei, Qi Han, Zhen Li, and Ming-Ming Cheng. Delving deep into label smoothing. IEEE Transactions on Image Processing, 30:5984–5996, 2021.

[179] Pengfei Zhang, Cuiling Lan, Junliang Xing, Wenjun Zeng, Jianru Xue, and Nanning Zheng. View adaptive neural networks for high performance skeleton-based human action recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 41(8):1963–1978, 2019.

[180] Songyang Zhang, Xiaoming Liu, and Jun Xiao. On geometric features for skeleton-based action recognition using multilayer lstm networks. In IEEE Winter Conference on Applications of Computer Vision, pages 148–157, 2017.

[181] Xikang Zhang, Yin Wang, Mengran Gou, Mario Sznaier, and Octavia Camps. Efficient temporal sequence comparison and classification using gram matrix embeddings on a riemannian manifold. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4498–4507, 2016.

[182] Yimeng Zhang, Xiaoming Liu, Ming-Ching Chang, Weina Ge, and Tsuhan Chen. Spatio-temporal phrases for activity recognition. In Proceedings of the European Conference on Computer Vision, pages 707–721. Springer, 2012.

[183] Zhengyou Zhang. Microsoft kinect sensor and its effect. IEEE Multimedia, 19(2):4–10, 2012.

[184] Zheng Zhang, Dazhi Cheng, Xizhou Zhu, Stephen Lin, and Jifeng Dai. Integrated object detection and tracking with tracklet-conditioned detection. arXiv preprint arXiv:1811.11167, 2018.

[185] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. In arXiv preprint arXiv:1904.07850, 2019.

[186] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krahenbuhl. Bottom-up object detection by grouping extreme and center points. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 850–859, 2019.

[187] Alex Zihao Zhu, Dinesh Thakur, Tolga Özaslan, Bernd Pfrommer, Vijay Kumar, and Kostas Daniilidis. The multivehicle stereo event camera dataset: An event camera dataset for 3d perception. IEEE Robotics and Automation Letters, 3(3):2032–2039, 2018.

[188] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Unsupervised event-based learning of optical flow, depth, and egomotion. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 989–997, 2019.

[189] Wentao Zhu, Cuiling Lan, Junliang Xing, Wenjun Zeng, Yanghao Li, Li Shen, and Xiaohui Xie. Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 30, 2016.

[190] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. Flow-guided feature aggregation for video object detection. In Proceedings of the IEEE International Conference on Computer Vision, pages 408–417, 2017.