Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza
1542

Reconocimiento de gestos por Cámara de eventos con técnicas de *Deep Learning*

Autor/es:
Joaquín González Oller

Director/es:
Ana Cristina Murillo Arnal
Iñigo Alonso Ruiz

Ingeniería Informática
especialidad Computación

Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

23 de Septiembre de 2020

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe entregarse en la Secretaría de la EINA, dentro del plazo de depósito del TFG/TFM para su evaluación).

D./Dª. Joaquín González Oller ,en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza, Declaro que el presente Trabajo de Fin de (Grado/Máster) Ingeniería Informática (Título del Trabajo)

Reconocimiento de gestos por Cámara de eventos con técnicas de Deep Learning

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 23 de Septiembre de 2020

Fdo: 

Resumen

Las cámaras de eventos son sensores de visión bio-inspirados cuya salida muestra cambios en la intensidad luminosa de la escena en vez de las imágenes RGB estándares de las cámaras tradicionales. Estas cámaras ofrecen grandes ventajas tales como un gran rango dinámico, no tienen distorsión por movimiento y su latencia de procesado es de microsegundos. Estas características hacen que sea una tecnología muy prometedora para diversas aplicaciones en el ámbito por ejemplo de la robótica o la vídeo-vigilancia. En particular este trabajo se centra en estudiar como utilizar este tipo de cámaras para reconocimiento de acciones, ya que potencialmente estas cámaras pueden ofrecer ventajas como ser capaces de captar movimientos a alta velocidad y con baja iluminación. Todavía hay pocos trabajos e investigaciones sobre esta aplicación de la tecnología de eventos, y por tanto el trabajo se centra en los siguientes puntos:

- Comprobar el funcionamiento de los métodos estándares para el reconocimiento de acciones en imágenes con las cámaras de eventos.
- Proponer métodos o mejoras sobre estos métodos tradicionales tanto en la representación de eventos como en el procesamiento de los eventos.
- Realizar pruebas de reconocimiento de acciones en escenarios donde cámaras tradicionales tienen problemas para obtener información significativa.

En este proyecto, en primer lugar se ha estudiado el funcionamiento de esta tecnología y datos y sistemas existentes para reconocimiento de acciones, con datos de eventos o imagen convencional. A continuación se ha diseñado, implementado y evaluado un sistema para reconocimiento de acciones a partir de información de eventos. Las fases principales de este sistema son las siguientes.

La **codificación de los eventos** en *frames*, donde nos centramos en evaluar dos representaciones, que se proponen en la literatura actual disponible: representación de eventos *por tiempo* y *por eventos*.

Un **clasificador** que predice la acción dado uno o varios *frames* de eventos. En particular se han implementado dos variaciones del sistema, con un clasificador que evalua los frames de forma individual y otro que clasifica grupos de frames.

Preprocesado y postprocesado de los frames. Se han propuesto dos estrategias de preprocesado de los *frames* antes de pasar al clasificador, y dos métodos de postprocesamiento para conseguir una predicción final más robusta, uno simple de consenso entre *frames* y otro de consenso ponderado buscando un método que posiblemente se adapte mejor a la evolución del movimiento.

Distintas configuraciones de estas fases se han evaluado sobre un modelo de red neuronal sencilla, que se establecerá como una red de arquitectura base para tener resultados de manera rápida y poder sacar conclusiones. La configuración que da mejores resultados, se ha evaluado de manera más exhaustiva con una arquitectura de red neuronal mucho más compleja, una red *Resnet50V2*, para ver mejor el posible alcance de los resultados.

Como principal resultado de este trabajo, se ha conseguido proponer un sistema adaptado a los datos de eventos que mejora el rendimiento respecto a procedimientos estándar para procesado de imagen convencional. En particular, se ha concluido que la representación de eventos *por eventos* es más robusta y mejor que la de *por tiempo*, debido a que no muestra las inconsistencias en la ejecución de movimientos, y que es esencial incorporar información de la evolución del movimiento.

Índice general

Capítulo 1

Introducción

Esta sección introduce la motivación y el contexto del proyecto, así como los objetivos a cumplir y las tareas realizadas.

1.1. Motivación y Contexto

Actualmente, se están utilizando técnicas de visión por computador en infinidad de aplicaciones como reconocimiento de acciones, reconstrucciones 3D y reconocimiento de objetos móviles. Estas aplicaciones utilizan técnicas para adquirir, procesar, analizar y comprender las imágenes del mundo real, con el fin de obtener información simbólica que pueda ser tratada por un ordenador para alcanzar cierto objetivo.

En la literatura de visión por computador y aprendizaje automático encontramos muchos métodos e infinidad de tecnologías que operan sobre cámaras convencionales. Estos métodos tienen el fin de enseñar a una computadora a entender como interpretar la información visual del mundo que le rodea, dotando al ordenador de la capacidad incluso de poder entender las interacciones entre personas o las actividades que realizan. En la Fig. 1.1 se muestran ejemplos de aplicaciones que utilizan el reconocimiento de acciones.

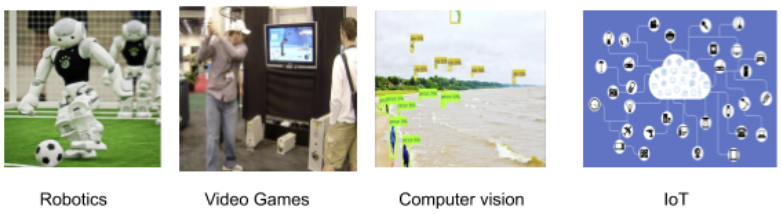


Figura 1.1: Campos de aplicación para el reconocimiento de acciones.

En este trabajo se explora una nueva tecnología, las cámaras de eventos, para realizar este tipo de tareas de reconocimiento. Este tipo de cámaras es una tecnología con acceso limitado a unos pocos prototipos, los pioneros en los que se centra este trabajo son de la firma *iniVation* [1]: *DVSExplorer*, *Davis346* y *DVSExplorer* Lite. Este tipo de cámara presenta propiedades muy beneficiosas y prometedoras, entre otras, para el tipo

[1] `https://inivation.com/`

de aplicaciones mencionadas que necesitan un reconocimiento de acciones automático, la baja latencia, gran rango dinámico y la ausencia de información del exterior puesto que solo captan la moción. Esta tecnología se describe en mas detalle en el capítulo 2. En la Fig. 1.2 se puede apreciar un ejemplo de la salida generada por las cámaras de eventos. Este output presentado en un formato especifico del *stream* de eventos.

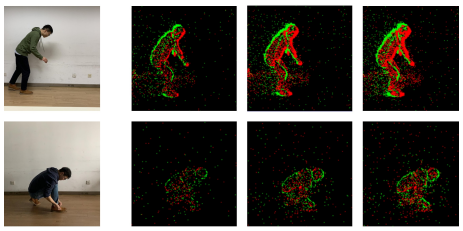


Figura 1.2: Imágenes en RGB (primera columna) e imágenes representando el *stream* de eventos (segunda, tercer y cuarta columna).

El principal reto de este trabajo se encuentra en estudiar los métodos existentes para la detección de personas y reconocimiento de gestos, seleccionando y adaptando los más adecuados para trabajar con los datos producidos por las cámaras de eventos. Las tareas y objetivos a realizar así como discusión de trabajos relacionados son detallados en las siguientes secciones.

Contexto del proyecto. Este proyecto ha sido realizado en el marco de una nueva línea de investigación que utilice cámaras de eventos, en el grupo de Robótica, Percepción y Tiempo Real del Departamento de Informática e Ingeniería de Sistemas de la Universidad de Zaragoza. A este grupo pertenecen los dos directores del trabajo.

1.2. Objetivos y Tareas

El objetivo principal de este proyecto es construir un sistema para reconocimiento de acciones con una cámara de eventos. Los problemas abordados son el uso de este tipo de cámaras, sus datos y cómo adaptar algoritmos de aprendizaje automático al reconocimiento de acciones con estos datos.

Para conseguir completar este objetivo, se han realizado las siguientes tareas (su extensión temporal resumida en el diagrama de la Fig. 1.3):

- T1. Prueba e instalación del software necesario para utilizar la cámara de eventos (*DVS-DAVIS346*). Se ha instalado el software en los distintos dispositivos de trabajo, estudiado su funcionamiento y verificado como realizar una grabación y una extracción de datos.
- T2. Estudio de los conceptos básicos sobre *Convolutional Neural Networks* (CNN) e instalación del software necesario para utilizar bibliotecas comunes de *deep learning* (*Tensorflow*, *Keras*). Este es el primer proyecto donde hacía uso de estas herramientas y tipo de redes por lo que se consultó documentación e información sobre estas para comprender su funcionamiento y utilización.
- T3. Estudio de métodos existentes, puesta en marcha de los más adecuados y evaluación de los mismos con datos públicos de cámaras de eventos. Más concretamente, esta tarea se divide en los siguientes puntos:

  - Estudiar y comparar los distintos métodos de extracción y representación de datos de cámaras de eventos.
  - Adaptar y poner en marcha dos sistemas de reconocimiento de acciones para utilizar las representaciones de eventos estudiadas anteriormente.
  - Estudiar las ventajas e inconvenientes de las representaciones con ayuda de los resultados obtenidos.

- T4. Propuesta e implementación de modificaciones o mejoras y evaluación de las mismas. Esta tarea consiste en aumentar y mejorar lo expuesto en la tarea anterior.
- T5. Captura de datos reales y evaluación de los métodos anteriormente implementados sobre los nuevos datos recogidos.
- T6. Redacción de documentación de los estudios, experimentos y software realizado.

	MARZO	MAYO	JUNIO	JULIO	AGOSTO
T1	10 horas				
T2		56 horas			
T3		60 horas	56 horas		
T4			64 horas		
T5				10 horas	
T6		8 horas	8 horas	8 horas	20 horas

Figura 1.3: Cronograma que representa la ejecución de las distintas tareas a lo largo del proyecto, asi como las horas aproximadas dedicadas a cada una de ellas.

1.3. Outline

El resto de este documento contiene, en el capítulo 2, la descripción en detalle sobre las cámaras de eventos: funcionamiento, procesamiento de datos, etc... En el capítulo 3, se describen los detalles sobre metodologías existentes y escogidas para el reconocimiento de acciones, así como las modificaciones propuestas: reconocimiento en imágenes de eventos por tiempo, por número de eventos o por paquetes de *frames*. En el capítulo 4 explica a grandes rasgos las configuraciones propuestas para ser evaluadas. El capítulo 5 resume los experimentos realizados y la evaluación de cada paso del sistema diseñado y el capítulo 6 resume las conclusiones obtenidas y posibles lineas de trabajo futuro.

Capítulo 2

Cámara de eventos

Las cámaras de eventos son sensores que captan cambios de luminosidad local. Estas cámaras no capturan imágenes en formato de intensidad de colores sino que reportan cambios de luminosidad en ciertos píxeles. En este capitulo se explica en detalle el funcionamiento de las cámaras de eventos, los datos de salida que generan así como sus ventajas y diferencias con respecto a las cámaras convencionales, un ejemplo siendo la Fig 2.1.

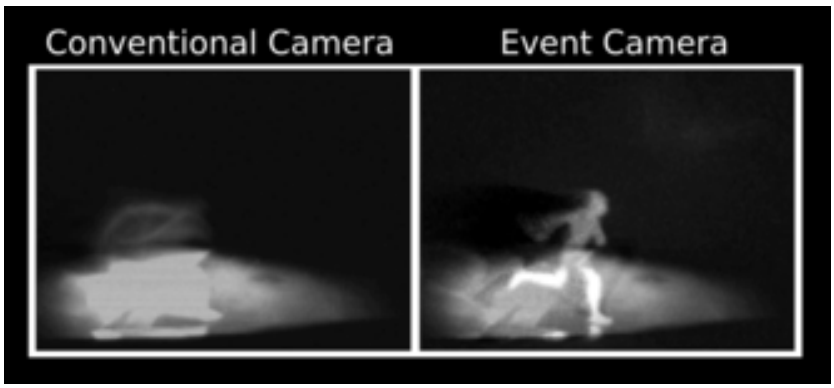


Figura 2.1: Captura en la misma escena con poca iluminación desde una cámara convencional, que apenas captura información, frente a una cámara de eventos.

2.1. Funcionamiento y Propiedades

Las cámaras de eventos son sensores bio-inspirados que funcionan de forma muy diferente a las cámaras tradicionales [1]. Mientras que las cámaras tradicionales capturan imágenes a un ritmo fijo y en general recogen información en todos los píxeles del sensor en cada una de las imágenes, las cámaras de eventos miden de forma asíncrona, y con una resolución de microsegundos, los cambios de luminosidad que ocurren en cada píxel (si es que ocurre alguno). Estas cámaras generan un evento cada vez que un píxel detecta un cambio de luminosidad. Estos cambios son generalmente provocados por movimiento, bien de la cámara, o de los elementos de la escena, aunque también

pueden ser generados por otro tipo de acciones, cambios en la intensidad de la escena. Los eventos que se generan tienen la siguiente información: *timestamp*, coordenadas del píxel, polaridad del evento (signo), es decir, si hay un incremento o decremento en la luminosidad.

Debido a esta forma de funcionar las cámaras de eventos presentan muchas propiedades interesantes entre las que cabe destacar:

Gran rango dinámico. En fotografía esto se describe como el ratio entre intensidad de luz máxima y mínima medible. En una cámara normal es 60 dB mientras que en esta es de 140 dB. Esto permite que en las cámaras de eventos incluso en escenas con iluminación baja aun se puedan capturar eventos haciendo posible poder realizar tareas de reconocimiento en escenas con condiciones de iluminación extremas.

Resolución temporal alta. Debido a que la salida a generar y el input a capturar es mucho más simple, la latencia entre la producción del evento y su captura es baja suponiendo una velocidad de computo mayor que en el caso de cámaras tradicionales. Siendo que las cámaras de eventos tienen tiempos de medida y respuesta de $1MHz$ mientras que las convencionales tan solo $1kHz$ máximo. Esto provoca que el reconocimiento se pueda producir a una velocidad mayor suponiendo una mejora en aplicaciones como conducciones autónomas de distintos objetos móviles y pueda ser utilizado en escenas con objetos dinámicos de gran velocidad.

Bajo consumo energético. De nuevo debido a su funcionamiento más simple y a su entrada asíncrona el consumo energético de estas cámaras es mucho menor. Esto posibilita que el hardware no se sobrecargué tanto y un beneficio para tecnologías que incorporen dichas cámaras.

No sufren de desenfoque por movimiento. Se entiende por desenfoque en movimiento como el rastro dejado por los objetos en movimiento en una fotografía o en una secuencia de imágenes. Esto se debe a limitaciones por la velocidad de obturación, velocidad de disparo, debido a que en estas cámaras la captura de eventos es muy rápida no se sufre de desenfoque.

2.2. Procesamiento de datos de cámaras de eventos

La salida que genera este tipo de cámaras no es la habitual. Las cámaras tradicionales generan una salida en un intervalo fijo de tiempo como se resume en la Fig. 2.2.

Figura 2.2: La captura de imágenes por una cámara tradicional se realiza en intervalos fijos. Lo cual supone, que los *frames* se generan de forma síncrona, como dados por una señal de reloj constante. Ejemplo cada segundo se realiza una captura de una escena.

En contraste, las cámaras capturan eventos de manera asíncrona. Un evento es generado cada vez que un píxel detecta un cambio de valor en la intensidad luminosa de la escena Fig. 2.3.

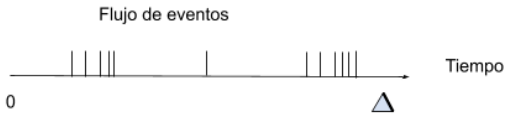


Figura 2.3: Como se puede apreciar las señales de captura se producen independientes del tiempo. Esto indica que depende de cuando suceda el evento la captura tendrá lugar. El conjunto de eventos se conoce como flujo de eventos, el cual se mantiene durante un cierto intervalo de tiempo.

Las cámaras de eventos consideran la intensidad luminosa en cada pixel como:

$$\pm C = \log I(x,t) - \log I(x, t - \Delta t), \tag{2.1}$$

donde $I(x,t)$ es la intensidad en un punto en un momento concreto , y definen un evento como:

$$event : (t, (x,y), sign(\frac{dI(x,y)}{dt}))$$

donde t es el *timestamp* en que el evento fue capturado, (x,y) son las coordenadas del píxel que capturó ese cambio de luminosidad, y $sign$ representa la polaridad (signo) del cambio de luminosidad, es decir, de la derivada de la intensidad de luminosidad respecto al tiempo en ese píxel.

Toda esta información capturada por el sensor se almacena un fichero, por ejemplo en formato AEDAT (Address Event DATA), que contiene una lista de eventos detectados desde que se activa la cámara hasta que es apagada o es detenida. Más detalles de este formato y el almacenamiento de los datos en el Apéndice A.

2.2.1. Procesamiento y representación de los eventos.

Es posible visualizar la salida de las cámaras de eventos como una imagen que resume los eventos detectados en escena. Los eventos que están dentro de la visión de la cámara, cada uno tiene las coordenadas en el espacio de la cámara. Se puede visualizar en este espacio 2D como una imagen.

Encontramos varias opciones sobre como construir estas representaciones en la literatura reciente [2]. Lo mas habitual es representar agrupaciones de eventos dentro de un periodo de tiempo establecido, como se resume en el Algoritmo 1, o juntando un número fijo de eventos, como representa el Algoritmo 2. El intervalo o número fijo de eventos a escoger dependerá de los parámetros con los que se hizo la grabación: velocidad del movimiento, repeticiones, etc.

Ambas estrategias tienen sus beneficios y sus problemas que se discutirán en secciones posteriores en más detalle. A continuación se describe un pequeño ejemplo ilustrativo. Imaginemos el siguiente escenario: Una persona saluda a la cámara. El movimiento será detectado por la cámara al producir un cambio de luminosidad. Los eventos serán registrados en el fichero aedat con la información anteriormente explicada. Y los dos tipos de imagen generada en este ejemplo serian los siguientes:

Algorithm 1: Generar un *frame* por tiempo

```
Result: frame
event = list frames(i);
intervalo = 2;
i = 0;
desde = event.timestamp;
fin = desde + intervalo;
while event.timestamp ≤ fin do
    frame[event.x, event.y] = event.polarity;
    i++;
    event = list frames(i);
end
```

Algorithm 2: Generar un *frame* por n eventos (n = 10)

```
Result: frame
event = list frames(i);
n = 10;
i = 0;
while i ≤ n do
    frame[event.x, event.y] = event.polarity;
    i++;
    event = list frames(i);
end
```

- Por tiempo (Algorithm 1): Se extrae la lista de eventos generados y por *timestamp* se van agrupando. Se cogen los eventos conforme a un rango de tiempo menor o igual a la duración de la grabación. Así se generarían N *frames* que recogen los eventos extraídos en un tiempo t. Supongamos que la grabación dura 10 segundos y agrupamos eventos sucedidos en una ventana temporal de 2 segundos tendríamos 5 *frames* de 2 segundos cada uno. El número de eventos variaría en cada uno pero la ventana temporal sería la misma.

- Número de eventos (Algorithm 2): La idea aquí es la misma pero generando N *frames* con el mismo número de evento m, siendo este menor o igual al número total de eventos obtenidos por la cámara. Supongamos que la grabación tiene 1000 eventos en total y queremos generar *frames* de 200 eventos cada uno. Tendríamos 5 *frames* con el mismo número de eventos cada uno mostrando la progresión temporal lineal de la grabación.

Cada forma de generar *frames* tiene sus ventajas y desventajas. En el caso del método por número de eventos, pierdes información temporal sobre los sucesos puesto que si se producen dos acciones diferentes de forma consecutiva estas podrían solaparse en un mismo *frame*.

Sin embargo el método por tiempo, el problema es que esta representación no es invariante a la velocidad de los movimientos. Movimientos rápidos van a generar *frames* con muchos eventos y movimientos lentos con pocos, aunque sean la misma acción y el mismo movimiento, la imagen o representación es totalmente diferente. Otro problema

con el método por tiempo, es que en el peor caso se tienen *frames* donde todos los píxeles son negros. Esto se debe a que en ese periodo de tiempo de la grabación no se han producido eventos o en caso contrario si hay mucho movimiento se verían demasiados eventos. Estos factores afectan al entrenamiento de manera negativa. Ejemplos de estos casos Fig 2.6.

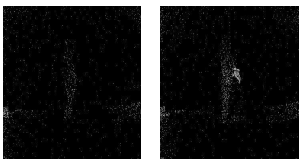


Figura 2.4: t=1200 us, t=13500 us

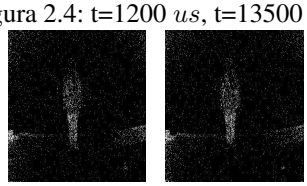


Figura 2.5: t=1200 us, t=13500 us

Figura 2.6: Imágenes generadas con distintas ventanas temporales, de menos a más de izquierda a derecha. Se puede apreciar que el primer par de imágenes pertenece a una grabación realizada por el método de tiempo 2.4 lo cual muestra una foto inicial del movimiento contra una en el punto intermedio del movimiento. Mientras que en la otra vemos el mismo caso con el método por número de eventos 2.5.

2.3. Ejemplos de Aplicaciones

Como se ha comentado anteriormente, este tipo de cámaras ofrecen una serie de ventajas, ya explicadas anteriormente, las cuales hacen de esta tecnología adecuada para numerosos campos de aplicación. En los últimos años encontramos propuestas para aplicar en campos tales como los siguientes.

Estabilización de vídeo. Las cámaras de eventos tienen píxeles que responden de forma independiente y asíncrona a cambios de luminosidad con una resolución de microsegundos. Gallego y Scaramuzza [3] tratan de establecer un método capaz de estimar la velocidad angular de la cámara. Uno de los métodos propuestos es la maximización del contrates. Este método ha sido demostrado capaz mejores resultados y mejores medidas giroscópicas en presencia de mociones de alta velocidad en comparación con cámaras tradicionales.

6DoF Tracking from Photometric Map. Al ser cámaras que no sufren de *blur* por moción, esto permite que sean dispositivos que den información consistente y confiable sobre mociones a alta velocidad o escenarios con baja luminosidad. Con estas propiedades en mente, encontramos trabajos como [4] que tratan de resolver el problema de realizar *tracking* con baja latencia con una cámara de eventos desde un mapa profundo

fotométrico construido mediante *pipelines* clásicos densos de reconstrucción. Se demuestra que gracias a las cámaras de eventos el método funciona en escenarios con moción de alta velocidad, que no sería posible con cámaras estándares.

3D Reconstruction with an Event Camera in Real-Time. En contraste con métodos tradicionales que intentan solucionar el problema estimando estructuras densas en 3D a través de un conjunto de *viewpoints*, se propone para cámara de eventos estimar una estructura 3D semi-densa con una trayectoria conocida. Una de las soluciones propuestas intenta explotar estas dos propiedades de las cámaras de eventos:

- La habilidad para responder a los bordes de la escena, que proporcionan información geométrica semi-densa sin necesidad de ningún tipo de pre procesamiento.
- El hecho de que proporcionan medidas constantes de la escena con el movimiento del sensor.

En [5] se detalla que las cámaras de eventos logran una mejora sobre los elementos de SLAM ya usado en métodos convencionales debido a las propiedades de estas. Provocando que escenarios inaccesibles, puedan ser atacados.

Motion Segmentation. Debido a que, los eventos son generados por la moción registrada en la escena, esto permite que la información registrada por las cámaras pueda encajar de forma más natural en la segmentación de la moción, especialmente en mociones de alta velocidad. Aquí se presenta la división de una escena entre diferentes objetos en movimiento [6].

La metodología propuesta permite una segmentación que muestra una mejora en resultados del 10 % sobre los métodos tradicionales. Este método usa una asociación entre evento y objeto que lo genera conociendo que la relación entre píxel y objeto puesto que cada píxel genera un evento tan solo si el objeto se mueve en la coordenada y los parámetros de la moción del objeto.

Drone Dodging Dynamic Obstacles. Actualmente se estudian metodologías para ver la rapidez de respuesta de un dron para esquivar objetos que aparecen como obstáculos de forma repentina. Estos métodos se sirven de la baja latencia que supone la generación del evento tras la captura sin necesidad del uso de otros sensores tal vez de mayor latencia y consumo eléctrico [7].

El método explota la información temporal que proporciona el flujo de eventos para poder distinguir de manera más rápida entre los elementos estáticos y móviles en una escena y busca esquivar de la manera más rápida aquellos objetos que se acercan al dron.

Todos estos estudios suponen un *baseline*, un comienzo, para futuros estudios sobre conducción autónoma o pilotaje autónomo usando estas cámaras.

Autonomous Drone Navigation in Low Light Lo mismo que lo anteriormente explicado pero realizando un estudio en circunstancias donde la iluminación es muy baja y las cámaras tradicionales fallan. Para así permitir una mejor conducción autónoma registrando simplemente los eventos generados y detectando así las ruta y los objetos en medio. Todo esto posible gracias al gran rango dinámico de las cámaras de eventos.

En este estudio [8] se aboga por un modelo híbrido que ha conseguido una mejora de precisión del 130 % sobre el uso de un pipeline solo de cámaras de eventos y del 85 % sobre *pipelines* de cámaras tradicionales .

Capítulo 3

Reconocimiento de acciones

Este capítulo discute los métodos existentes para tareas de reconocimiento automático de acciones, así como las modificaciones propuestas para realizar reconocimiento de acciones con las cámaras de eventos.

3.1. Introducción

El reconocimiento de acciones es una tarea que consiste en identificar que acción se está realizando, de entre un conjunto de posibles acciones, a partir de unos datos de entrada, que en nuestro caso es un vídeo capturado con una cámara de eventos. Este problema se puede modelar de dos formas principalmente: (1) tratando las diferentes imágenes como una secuencia y, (2) tratando las imágenes de forma independiente para después realizar un consenso entre las diferentes predicciones independientes. Para esta tarea se encuentran varios retos que resolver. Sobre todo cabe destacar los siguientes:

- Gran coste computacional. Los datos a procesar ya tienen una gran extensión siendo de NxM píxeles. En el caso de un vídeo se tienen N imágenes de NxM píxeles. Todas estas deben ser estudiadas, analizadas por una red. Lo cual provoca que cuanto más compleja sea la red el tiempo de computo aumentará. Cuanto mayor el conjunto de datos mayor será el coste temporal.
- Capturar un largo contexto. El reconocimiento de acciones supone la captura de un contexto espacio-temporal a lo largo de varios *frames*. Además, la información espacial puede no ser capturada al detalle por una cámara, omitiendo detalles finos, clave a la hora de poder clasificar una acción. Un ejemplo sería intentar clasificar dos movimientos similares como es nadar a *crawl* o a braza, cuya diferencia radica tan solo en unos momentos puntuales.
- Diseñar una arquitectura de reconocimiento. Diseñar una arquitectura que pueda capturar la información espacio temporal engloba múltiples opciones no triviales y costosas de evaluar. Algunas posibles estrategias pueden ser:
    - Una red para capturar información espacio-temporal vs una para capturar información espacial y otra para temporal
    - Fusionar predicciones a través de diferentes subconjuntos de imágenes.

  - *End-to-end training* vs una clasificación independiente por *frame*. Una estrategia es obtener una predicción/clasificación de toda la secuencia de vídeo. O en cambio se podría hacer la opción de obtener resultados sobre cada *frame* de forma individual y posteriormente procesar dichos resultados para clasificar la acción realizada en el vídeo.

- No hay un *benchmark* estándar. No es fácil encontrar una base de datos estándar y reconocida sobre la cual realizar este tipo de reconocimiento, y aun menos para el reconocimiento de acciones en cámaras de eventos. Además cada base de datos puede ser buena para el reconocimiento de acciones dentro de un contexto pero no de forma global entendiendo que la misma acción en diferentes entornos tiene connotaciones diferentes.

3.2. Trabajos relacionados

3.2.1. Reconocimiento de acciones con cámaras convencionales

Antes del *boom* en el uso del *deep learning*, la gran mayoría de los algoritmos para el reconocimiento de acciones pueden ser divididos en tres fases:

- Extracción de características que describen una región del vídeo: de forma densa [9] o utilizando un conjunto de puntos locales de interés [10][11].
- Las características extraídas se combinan en una descripción a nivel de vídeo de tamaño fijo. Una variante popular es usar una bolsa de palabras visuales para codificar las propiedades [12].
- Un clasificador es entrenado sobre el descriptor de tamaño fijo, por ejemplo la bolsa visual de palabras, para obtener la predicción final.

Después de 2014, ya con la explosión del uso de *deep learning* en visión por computador, se pueden encontrar dos grandes tipos de arquitecturas de redes utilizadas para el reconocimiento de acciones. La diferencia entre ambas se encuentra en la forma de combinar la información espacio-temporal.

Red con un único *stream*. Este método busca explorar múltiples maneras de fusionar información temporal de *frames* consecutivos pudiendo usar convoluciones o no. Incluso siendo estas pre-entrenadas o no.

Como se puede ver en la Fig. 3.1, el input consiste de los *frames* generados en un vídeo. Single *frame* o *frame* único, usa una sola arquitectura que fusiona la información de todos los *frames* en el ultimo paso. *Late* fusión o fusión tardía usa dos redes con parámetros en común y también combina predicciones al final. *Early* fusión o fusión temprana consiste en realizar una convolución de n-*frames*. *Slow* fusión consiste en fusionar *frames* a varios niveles, un equilibrio entre *early* y *late* fusión.

Después de extensas experimentaciones, [13] encontró que los resultados eran significativamente peores en comparación con los métodos tradicionales anteriormente explicados y que hasta el momento se realizaban. Los motivos eran los siguientes:

- Las características espacio-temporales no capturaron características de moción.
- Un conjunto de datos poco diverso provocaba que el aprendizaje de estas características detalladas fuera costoso.

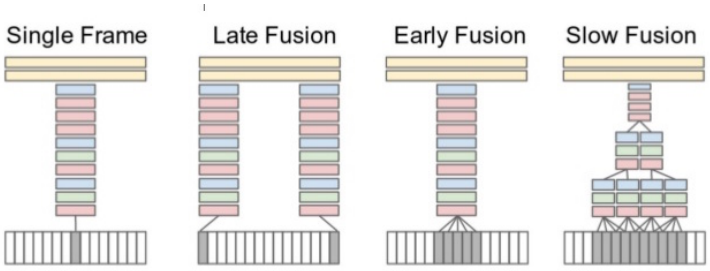


Figura 3.1: Métodos usados para una red con un único *stream*

Red con dos *streams*. Este método, Fig. 3.2 consiste en tener dos redes separadas una conteniendo el contexto de moción y otra con el contexto de espacio [14]. El input de la red espacial es un único *frame*. Para la otra red se decidió introducir como datos un flujo óptico *multi-frame*. Las dos redes son entrenadas de forma separada y luego combinadas. La predicción final fue calculada de la misma forma que en el método anterior. Aunque este método conseguía resultados mejores resultados que el método de red con un único *stream* aun decían los investigadores que tenía ciertos problemas:

- Aunque las etiquetas eran predichas realizando una media entre las diferentes predicciones de los *frames* de un vídeo, la información temporal a través del total del vídeo aun se perdía debido a que se tomaba tan solo conjuntos de *frames* dentro de un mismo clip.
- Los *frames* de un vídeo eran sampleados de forma uniforme, provocando así que el *ground truth* de todos estos sen el mismo a pesar de que el movimiento tan solo tome unos segundos del vídeo.
- Es un método que exige pre procesamiento de imágenes para el cálculo de los vectores *multi-frame*, dos entrenamientos de dos redes por separado y luego la combinación de estos siendo costoso todo el proceso.

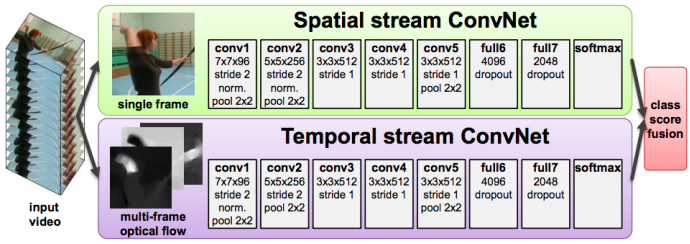


Figura 3.2: Muestra de como sería la arquitectura empleada en este método.

3.2.2. Reconocimiento de acciones con cámaras de eventos

Las cámaras de eventos son una nueva tecnología que está en plena evolución y experimentación de sus capacidades y límites. La mayoría de trabajos con este tipo

de cámaras se centran en reconocimiento de objetos o tracking [1]. Son pocos los trabajos los cuales se centran en el reconocimiento de acciones en vídeo. No obstante, si hay varios que hablan sobre como realizar representaciones apropiadas para tener conjuntos de datos en imágenes sobre la información obtenidas por las cámaras [15] o experimentos que en naturaleza son similares [16].

Los pocos trabajos que existen de reconocimiento de acciones de cámaras de eventos se centran en aspectos muy concretos de las cámaras de eventos, comprobando si dicha tecnología puede servir para el campo[17] [18]. En los trabajos se puede ver como enumeran retos que conllevan el uso de esta nueva tecnología que genera dificultades y sus posibles soluciones: el hecho del gran ruido que generan o la perdida de la información temporal dependiendo de la representación. En otros, se nos explica como las cámaras de eventos pueden intrínsecamente servir para capturar información clave de un movimiento, clave para su correcta clasificación y entendimiento.

Sin embargo lo que no hacen es comparar y estudiar como afectan diferentes representaciones de los eventos en esta tarea. Esto es uno de los análisis que se plantean en este trabajo. Al contrario que estos trabajos, planteamos pre procesados y post procesados interesantes sobre estos datos que ayuden a la tarea de reconocimiento de acciones. Otra cuestión importante que no se ve en otros trabajos es la comparación entre cámaras tradicionales y las de eventos. Suponen estas una mejora, aportan aspectos nuevos a esta área o por el contrario la ventana de captación y el alto procesamiento no suponen nada. Todo esto se abarca a lo largo de este trabajo a través de una serie de experimentaciones, apoyándose de los resultados para establecer las conclusiones.

Capítulo 4

Sistema propuesto

Este capítulo describe las dos principales variaciones implementadas para el sistema de reconocimiento de acciones a partir de datos de eventos desarrollado en este proyecto.

4.1. *Overview*

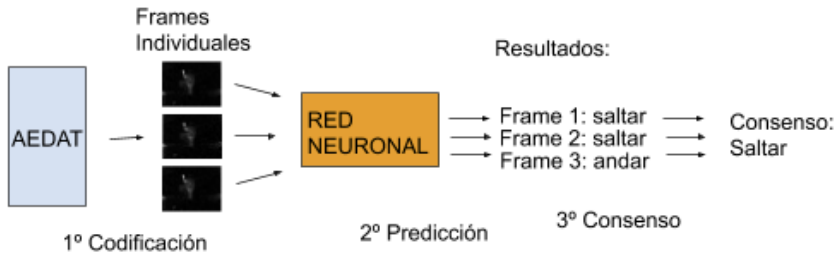
El objetivo del sistema es recibir como entrada una serie de *frames* y predecir la acción que esta ocurriendo en cada uno de los *frames*. La principal diferencia entre las dos variaciones implementadas recae en como se procesan los *frames* de eventos, si de manera individual, o por bloques, como se resume a continuación.

Método por *frames*. Se implementa un método que clasifica cada *frame* de eventos de manera independiente, para luego poder realizar un consenso entre los distintos *frames* pertenecientes al mismo vídeo para predecir la acción en el vídeo. El método consiste en las siguientes fases, que están resumidas en el diagrama de la Fig. 4.1:

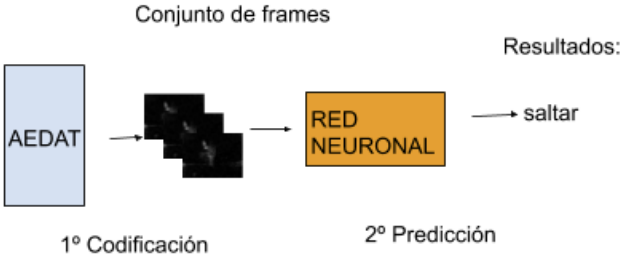
1. La codificación del output de eventos en *frames*.
2. Predicción de la acción para cada uno de los *frames* con la CNN entrenada. Esta CNN habrá sido entrenada previamente de manera supervisada a partir de un conjunto imágenes.
3. Consenso entre los *frames* para dar un resultado sobre la acción contenida en el vídeo.

Método por *conjunto de frames*. Se implementa un método para clasificar conjuntos de *frames*. Cada conjunto es un vídeo de un movimiento por lo que el resultado sería directamente la predicción sobre el vídeo. El método consiste en las siguientes fases que están resumidas en el diagrama de la Fig. 4.1:

1. La codificación del output de eventos en *frames*.
2. Predicción de la acción para el conjunto de *frames* con la CNN entrenada. Esta CNN habrá sido entrenada previamente de manera supervisada a partir de un conjunto imágenes.



(a)



(b)

Figura 4.1: Sistema propuesto para reconocimiento de acciones a partir de información eventos (fichero AEDAT). Las dos versiones diseñadas utilizan la información de los eventos como entrada a una red neuronal que predice la acción. La diferencia recae en como procesan los eventos: (a) con representación *por frame* (b) con representación *por conjunto de frames*.

4.2. Fases del sistema

Esta sección detalla las fases más importantes en el sistema propuesto y las diferencias en cada una de ellas a lo largo de las dos variaciones descritas. Aquí se explica a grandes rasgos los procedimientos empleados en dichas etapas así como los parámetros usados para cada una de éstas.

4.2.1. Codificación o Representación de los eventos

Como se ha mencionado anteriormente, la información de los eventos esta almacenada en un fichero de texto (formato AEDAT). En este fichero, se recoge información de las siguientes variables: coordenada x, coordenada y del píxel, *timestamp* del momento en el que se capturo el evento, y el signo de la polaridad. En el sistema propuesto tan solo se hace uso de las coordenadas x e y para conseguir la representación que se detalla a continuación.

El objetivo es codificar la información de los eventos en una matriz cuyas dimensiones son iguales a la resolución de la cámara: 346x260. Esta matriz almacena el número de eventos obtenidos en una coordenada. En otras palabras, si la cámara detecto un evento en el píxel (x, y) entonces la matriz en la coordenadas (x, y) tiene valor 1, si se

encuentran más eventos con coordenadas (x, y) entonces el valor aumenta según ese número de ocurrencias. Tras este paso, se normaliza esta matriz a un intervalo de [0, 255].

Para generar esta representación, como se ha explicado en el Cap. 2, hay dos estrategias que se usan frecuentemente. En este proyecto se han implementado ambas.

Con respecto al método de generación de imágenes explicado en el Cap. 2, en vez de codificar la polaridad de los eventos en cada píxel de la imagen generada, se codifica el histograma de eventos (cantidad de eventos).

En los experimentos se han utilizado estas dos representaciones:

- Para la representación por tiempo se ha implementado el Algoritmo 3. En ella se agrupan los eventos según intervalos de tiempo, en concreto intervalos de 10000 μs.
- Para la representación por eventos se ha implementado el Algoritmo 4. En ella se agrupan los primeros n eventos siendo ese n igual a 7000 eventos.

Algorithm 3: Generar la matriz de ocurrencias de eventos (*frame*) para la representación por tiempo (para un intervalo de 10000 μs.

```
input: all-events;
intervalo = 10000;
t = firstevent.timestamp;
list_events_in_interval = [find events in all-events with event.timestamp in
 [t,t+intervalo] ]
for event in list_events_in_interval do
    frame[event.x, event.y] += 1;
end
return frame;
```

Algorithm 4: Generar la matriz de ocurrencias de eventos (*frame*) para la representación por eventos (con n=7000).

```
input: all-events;
n = 7000;
i = 0;
event = list frames(i);
while i ≤ n do
    frame[event.x, event.y] += 1;
    i++;
    event = list frames(i);
end
return frame;
```

4.2.2. Pre procesado de las imágenes.

Tras generar los *frames* en el paso anterior, es necesario realizar un pre procesado para detectar automáticamente aquellas que no resulten útiles para el entrenamiento del

clasificador. Son muchas las imágenes generadas y la gran mayoría de los movimientos son muy similares, debido a que todos los usuarios parten de una misma posición inicial y final.

Sumado a esto, están las similitudes entre movimientos. Los movimientos tan solo se diferencian en las fases intermedias de la acción pero el resto de imágenes son casi idénticas. Esto provocaría resultados no deseables, en concreto un clasificador no robusto, si se utilizan todos los *frames* para entrenar el clasificador de acciones, por lo tanto es preferible eliminarlas.

Aquí de nuevo se tiene que distinguir entre los dos métodos de representación descritos en el paso anterior, ya que el pre procesado se ha implementado de manera diferente para cada uno de ellos:

Pre procesado para la representación por tiempo. Para cada uno de los *frames* anteriormente generados se calcula el número de eventos de dicho *frame*. Se calcula el mayor número de eventos max y el menor min de la grabación. Esto sirve para estimar cuales serían los *frames* más significativos y los menos significativos. Esto se debe a que las fases intermedias del movimiento tienen un mayor número de eventos. Solo las imágenes que cumplan que su número de eventos es mayor a $value$ son seleccionadas. El valor de $value$ se calcula como:

$$\begin{aligned} value = max - min \quad & if\ max \geq 5 * min \\ value = max - (min/div) \quad & if\ max < 5 * min. \end{aligned} \tag{4.1}$$

El motivo por el cual se opta por estas dos condiciones es el siguiente. Si la disparidad es muy grande entre el máximo y el menor entonces es que la gran mayoría de los *frames* son no significativos o en otras palabras no demuestran movimiento significativo, por tanto no serían imágenes pertenecientes a las fases intermedias del movimiento.

Mientras que si es el caso contrario, hay que tener cuidado de no eliminar más de las deseadas. De ahí que se busque eliminar menos de las deseadas debido a que la disparidad entre max y min es pequeña, de ahí que min se divida por un valor div.

Pre procesado para la representación por eventos. Para la representación por eventos se realiza lo mismo pero con la diferencia que ahora todas las imágenes tienen el mismo número de eventos. Por tanto se eliminan las n primeras y n últimas imágenes de cada movimiento.

Procesado *por conjunto de frames*. Se realiza un proceso similar tal como se describe en el algoritmo 5. En este caso no se ha realizado una eliminación de *frames* puesto que se generan conjuntos de n *frames* que muestren la evolución del movimiento. Por lo tanto si se tiene un vector de 100 imágenes de un movimiento. Se intentan extraer n imágenes que muestren las fases iniciales, intermedias y finales. Por lo que se agrupan *frames* que se encuentren al inicio del vector, en el medio y al final.

4.2.3. Reconocimiento de la acción

Junto con la elección del conjunto de datos otro punto crítico en el sistema es la elección de la red neuronal más adecuada para este tipo de clasificación o de aprendizaje de acciones. Puesto que este trabajo hace uso de tecnología novedosa donde no disponemos de ejemplos anteriores, se plantea evaluar dos arquitecturas diferentes:

Algorithm 5: Implementación para la construcción de los conjuntos de *frames*.

```
Result: conjunto_frames
input:=n;
i = 0;
while i ≤ n - 1 do
    step = len(frames)/n;
    conjunto_frame[i] = frames[i * porcion];
    i++;
end
```

- Una primera red FC-base (FC, *fully-connected*) para obtener una primera impresión de la complejidad del problema.
- Una segunda red convolucional siguiendo una arquitectura estándar para clasificación de imágenes, para verificar si mejora los resultados.

Red *FC-base*. Esta red se basa en tomar los píxeles como elementos independientes entre ellos, pasándolos a un vector (quitando lo que es la relación entre vecinos) y las operaciones que se realizan son *fully connected*. La red tiene dos capas *fully connected* de tamaño 128 y 10. Esta red se usa simplemente para tener un primera experimentación orientativa simple y rápida para ver si los resultados son buenos o se necesita un gran trabajo y redes complejas para realmente ver que esta tecnología sea usada en los campos de *deep learning*.

CNN para clasificación de imágenes, arquitectura *ResNet50V2* [19] Es una red bien conocida y usada para este tipo de experimentos y aplicaciones [20]. La red *ResNet50V2* añade un modelo pre entrenado convolucional que añade complejidad al modelo. Esta capa es para ver que resultados obtenemos con una red más compleja que se usa en experimentos similares. Esta red también termina con dos capas *fully connected* (de tamaño 1024 y 10).

En ambas arquitecturas, las imágenes o grupos de imágenes son introducidos de forma aleatoria en grupos o conjuntos de imágenes, conocidos como *batches*, asociando sus etiquetas correspondientes. Se utilizan *batches* de 20 imágenes y realizándose un entrenamiento de 10 épocas (una época es un procesado completo del *data set*). La función *loss* usada en el entrenamiento es la *Cross-Entropy loss* :

$$CE = -\sum_{i}^{C} t_i log(s_i) \tag{4.2}$$

donde t_i y s_i son la instancia real o etiqueta real y la predicción obtenida para cada unas de las clases $_i$ en C. Esta función de *loss* permite medir la clasificación realizada por un modelo que muestra una probabilidad de etiquetado entre las clases. Esta función incrementa si la probabilidad de clasificación diverge de la etiqueta real.

4.2.4. Consenso final

Al final del entrenamiento se produce la clasificación o etiquetado de los distintos *frames* para cada vídeo. Es el listado de etiquetado realizado a cada uno de los *frames*,

es decir, de las imágenes de un vídeo por tanto no es el resultado final. Para ello hay que realizar un consenso entre estos *frames* para ver cual es la etiqueta más común y por tanto cual sería el movimiento correspondiente al vídeo. En este trabajo se consideran dos consensos diferentes.

Consenso general. Realizar un consenso o votación entre los *frames* de cada vídeo para así determinar cual es la etiqueta o movimiento más común entre los *frames* y este ser el resultado final. Evidentemente, este método tiene varios fallos y tan solo sirve como un punto inicial para ver como de bien un método simple funcionaría en el reconocimiento de imágenes. Los problemas están en que los puntos característicos de cada movimiento puede que tan solo duren por una fracción de tiempo. Esto provoca que en un consenso estos puntos tan importantes y claves para el etiquetado no tengan más peso que los demás y por tanto al ser en menor cantidad no determinen el etiquetado final de manera correcta.

Consenso ponderado. Para resolver el problema mencionado en el consenso general, este método, visible en la fig 4.2 busca encontrar estos *frames* intermedios y característicos del movimiento y darles mayor peso en la votación. Para ello el *data set* usado y antes explicado se ve que consiste tan solo de una repetición por movimiento. Esto permite realizar una especie de función gaussiana que aumente el peso de cada *frame* en la votación siguiendo lo que sería la progresión temporal del movimiento. De tal manera que el pico de la función sea el *frame* en el punto álgido e intermedio del movimiento y luego de nuevo decrece dando a las fases finales menor peso.

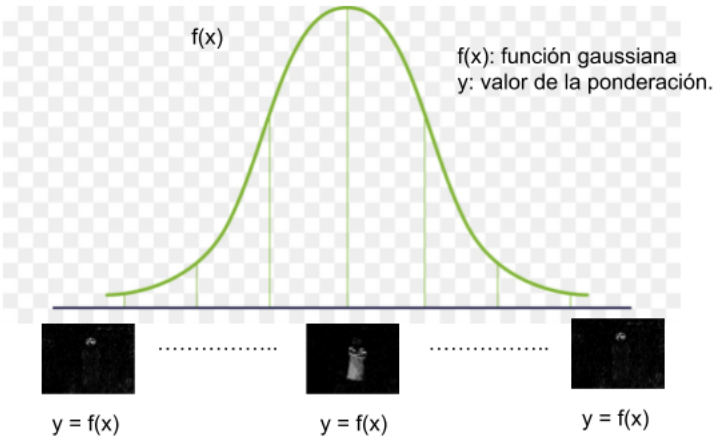


Figura 4.2: Explicación de la función implementada para realizar una votación ponderada. Como se puede ver, cada imagen tiene un valor asociado que sería el valor devuelto por la función gaussiana $f(x)$, este sería el peso que tiene dicho *frame* dentro de la votación.

Capítulo 5

Experimentos y evaluación

Este capítulo describe la evaluación realizada de los sistemas considerados y propuestos. El principal objetivo es validar los resultados para determinar si esta nueva tecnología es compatible con métodos existentes para el reconocimiento de acciones en la área de visión por computador, así como si las mejoras propuestas consiguen resultados óptimos.

5.1. Métricas y Datos utilizados

Esta sección describe los conjuntos de datos usados para el entrenamiento y validación de los modelos. También se describen las métricas utilizadas para esta evaluación.

5.1.1. Conjuntos de datos

***Data set Neuromorphic Benchmark* [15].** Este *data set* consiste de 3 partes: reconocimiento de acciones, reconocimiento de caídas y reconocimientos de viandantes. Para los experimentos se usa el conjunto de datos de la parte de reconocimiento de acciones Fig 5.1.

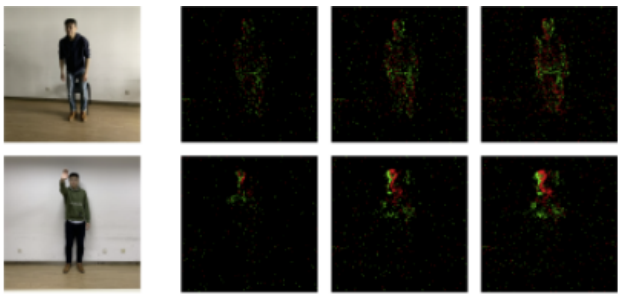


Figura 5.1: Muestra de dos acciones (arriba: levantarse, abajo: saludar) del *data set* [15]. A la izquierda un *frame* del vídeo convencional capturado simultáneamente a los eventos. A la derecha 3 *frames* de eventos correspondientes a cada acción.

Este *data set* está compuesto por vídeos realizados con 15 sujetos diferentes en un único escenario de oficina, la misma oficina en todos ellos. La longitud de los vídeos es de aproximadamente 5 segundos, unos 450 *frames* de media cada uno.

Este *data set* fue capturado con una cámara DAVIS346, mismo modelo que la disponible en este proyecto. Se recogen 10 acciones diferentes: Cruce de brazos, levantarse, dar una patada, coger algo del suelo, saltar, sentarse, lanzar, darse la vuelta, andar y saludar. En el apéndice C se muestra un ejemplo de cada una.

Aclarar que cada movimiento es realizado por cada sujeto una única vez, por lo tanto solo se tiene una repetición por movimiento. Además cada sujeto ha sido grabado tanto desde una perspectiva frontal como desde ambos perfiles.

Los motivos por los cuales esta base de datos fue elegida son los siguientes:

- Numero de movimientos. El número de movimientos no es muy extenso lo cual permite mirar los resultados fácilmente y poder extraer conclusiones sin ser demasiada información que procesar.
- Los movimientos son realizados una única vez. Esto facilita el aprendizaje y centrarse en el movimiento sin tener en cuenta factores como: rapidez de cada movimiento dentro de un vídeo, agrupación de *frames* muy rápidos solapando repeticiones, etc.
- Se hace uso de la misma cámara. La cámara de eventos es el mismo modelo que la usada para estos experimentos disponible en el laboratorio. Lo cual permite poder usar luego la cámara disponible y poder ver si se pueden replicar los resultados.
- Se realiza dentro de un escenario replicable. Se realizaron las grabaciones dentro de una oficina con una cámara grabando desde una mesa. Los sujetos se podrían de frente a la cámara o de lado.

***data set* propio de validación.** Para validar los resultados obtenidos y experimentos realizados se necesita ver como los modelos generados para la clasificación se comportan frente a datos generados por una grabación de una cámara bajo diferentes escenarios. Por lo tanto con la cámara de eventos disponible se replica el mismo escenario y *data set*. Pero esta vez cambiando la luminosidad y la velocidad de movimiento así como se busca ver que sucede si se toma todo siguiendo los mismos parámetros del *data set* anterior Fig 5.2.

Figura 5.2: Muestra de las grabaciones realizadas con la cámara de eventos propia.

Por lo tanto este conjunto de datos consiste de lo mismo que el anterior. No obstante en este caso se realizan bajo 2 escenarios diferentes: iluminación media e iluminación baja. Además en cada caso se realiza una grabación de los movimientos a velocidad alta y media. No se buscan escenarios con iluminación alta o velocidad baja puesto que estos escenarios no suponen cambio alguno sobre los de iluminación media y velocidad media. Puesto que lo interesante es comprobar con respecto a las cámaras tradicionales, si en escenarios donde estas no pueden operar esta nueva tecnología puede funcionar correctamente.

data set	**Nº Usuarios**	**Nº Escenarios**	**Acciones por usuario**	**Repeticiones por acción**
N. Benchmark[15]	15	1	10	1
Propio	2	1	5	1

Tabla 5.1: Resumen del contenido de los dos conjuntos de datos utilizados

5.1.2. Métricas

Todos los experimentos de este capítulo se evalúan con las siguientes métricas o cálculos:

Precisión. Es la fracción de instancias relevantes entre el número total de instancias obtenidas. En otras palabras es el número de predicciones correctas, o verdaderos positivos (VP) entre el número total de veces que se ha dicho que una instancia pertenecía a esa clase, es decir, la suma de todos los positivos para esa clase, verdaderos y falsos positivos (VP+FP).

$$Precision = \frac{VP}{VP + FP}. \tag{5.1}$$

En particular, en los experimentos se calcula la **precisión media por clase**, para tener una media equitativa entre las distintas clases independientemente del número de imágenes que obtenemos por cada movimiento. También se calcula la **precisión media por *frame*** la cual es una precisión general que no tiene en cuenta el número de imágenes por cada movimiento tan solo la media entre las clases.

***Recall*.** Es la fracción del total de casos dichos que pertenecen a una clase sobre el total de casos que son de una clase. En otras palabras es el número de predicciones correctas, o verdaderos positivos (VP) entre el número total de instancias de una clase, es decir, la suma de verdaderos positivos y falsos negativos (VP+FN).

Por lo que la formula matemática del *recall* sería:

$$Recall = \frac{VP}{VP + FN}. \tag{5.2}$$

En particular, en los experimentos se calcula la ***recall* media por clase**, para tener una media equitativa entre las distintas clases independientemente del número de imágenes que obtenemos por cada movimiento. También se calcula la ***recall* media por *frame*** la cual es una precisión general que no tiene en cuenta el número de imágenes por cada movimiento tan solo la media entre las clases.

Matriz de confusión. Una matriz de confusión es una herramienta que permite conocer que clases se confunden más las unas con las otras y por cuanto es. La matriz de confusión es una matriz que indica las instancias o clasificaciones para cada una de las clases.

La matriz por tanto tendría en las filas las instancias reales y en las columnas las clasificaciones realizadas. Cada una de las casillas por tanto indicaría para cada clase el número de veces que ha sido etiquetada a una de las clases.

Por ejemplo, si tenemos un problema de clasificación en tres clases, la matriz seria como muestra la Tabla 5.2, donde x_{ij} sería por tanto el número de predicciones de

Matriz de confusión	clase 1	clase 2	clase 3
clase 1	x_{11}	x_{21}	x_{31}
clase 2	x_{12}	x_{22}	x_{32}
clase 3	x_{13}	x_{23}	x_{33}

Tabla 5.2: Ejemplo de matriz de confusión.

clase j para elementos que realmente son de las clase i. Esta matriz será esencial para explicar problemas de distinción entre los movimientos y discernir la dificultad que tiene reconocer imágenes de acciones.

5.2. Experimentos

En los experimentos realizados se analiza el rendimiento de las distintas configuraciones para cada una de las representaciones estudiadas para los eventos capturados (*por tiempo* o *por número de eventos*):

- Sin-preprocesado: **Base**.
- Dos tipos de pre-procesado: **Pre1** y **Pre2**, como se explica en el capitulo 4, **Pre1** sería el preprocesado para la representación por tiempo y **Pre2** para la representación por eventos.
- Dos tipos de post-procesado: **Post1** (Votación) y **Post2** (Votación ponderada), explicados en la capitulo 4.

Debido a la extensión de los datos y el número de resultados aquí se muestran aquellos más importantes y trascendentes para alcanzar el objetivo del trabajo. Los resultados complementarios y detallados por clases se encuentran en el ApéndiceB.

También es importante destacar que en la mayoría de los experimentos, solo se ha evaluado la arquitectura simple o base (red FC-base). Esto se debe a que el entrenamiento con una red mas compleja tiene un alto coste temporal. Solo para la mejor configuración se re-evaluará utilizando la arquitectura mas compleja puesto que sería la que garantizaría mejores resultados.

5.2.1. Análisis de la representación de eventos *por tiempo*

En primer lugar, se analiza cual es la mejor configuración para este tipo de representación. La Tabla 5.3 muestra el resumen de los resultados para las distintas configuraciones evaluadas. Se muestran tanto las medias por clases como la de por *frames* como comparativa, para ver como afecta que cada movimiento no tenga el mismo número de imágenes.

Base. En este experimento no se hace uso ni de pre procesados ni post procesados, se usan todos los *frames* extraídos directamente de los ficheros y estos introducidos a la red FC-base. Este experimento sirve como una primera toma de contacto para entender como de bien funciona esta nueva tecnología. Además de servir para evaluar como de necesarias son las mejoras. Se puede comprobar como la precisión y *recall* por *frames* es bastante baja.

Tabla 5.3: Resultados (Precisión y *recall*) de la clasificación con variaciones de la representación *por tiempo*. Arquitectura de red *FC-base*.

	Precisión (avg. avg. per class)	*recall* (avg. avg. per class)	Precisión avg. per frame	*recall* avg. per frame
Base	0.27	0.23	0.25	0.23
Base+Pre1(div=2)	0.47	0.47	0.41	0.37
Base+Pre1(div=3)	0.51	0.54	0.42	0.39
Base+Pre1(div=3)+Post1	**0.55**	**0.61**	**0.46**	**0.45**
Base+Pre1(div=3)+Post2	0.44	0.43	0.37	0.29

Esto indica que hay serios problemas para entender las características del movimiento. Esto se debe a que los movimientos son bastantes similares entre ellos sobre todo en fases iniciales y finales, incluso algunos como levantarse o sentarse son acciones en si equivalentes solo realizadas al revés.

Además, las cámaras de eventos sacan cambios de luminosidad. Esto quiere decir que, al contrario de las cámaras tradicionales, las de eventos no dan información sobre el contexto o la escena. Tan solo sobre los cambios de luminosidad de una escena por lo que la acción de saltar se ve perjudicada, esto debido a que no se puede ver que el usuario esta más a arriba situado con respecto al suelo. Connotación importante si evaluamos los *frames* individualmente.

Base+Pre 1 (div=2). En este experimento se hace uso del pre procesado para la representación por tiempo con valor div 2, ningún post procesamiento y con la red FC-base. El objetivo es ver como eliminar aquellos *frames* erróneos y coincidentes entre acciones hace aumentar las medias de precisión y de *recall*.

Se aprecia como las medias se doblan. Esto sería indicación definitiva de la necesidad de filtrar las imágenes para eliminar todas aquellas que no sean deseables. Debido a que el número de *frames* coincidentes disminuye, la precisión del etiquetado aumenta así como el *recall*. Esto se debe a que no hay tantas imágenes que puedan ser o no clasificadas o clasificadas erróneamente en otra clase.

Base+Pre 1 (div=3). En este experimento se hace uso del pre procesado para la representación por tiempo con valor div 3, ningún post procesamiento y uso de la red FC-base. El objetivo es ver como un mejor o peor filtrado de las imágenes afecta a los resultados y conseguir una afirmación más de que las medias aumentan con un filtrado.

Las medias vuelven a aumentar. Esto refuerza el hecho de que un filtrado es más que necesario y que la codificación y construcción correcta de las imágenes es clave. La distinción entre los movimientos debe de ser más que clara o de lo contrario los valores son bajos.

Sin embargo, un filtrado muy agresivo sería contraproducente y que es clave conocer como de precisa es la eliminación de *frames* erróneos o de lo contrario habrá clases que serán irreconocibles. Además de que si son filtrados agresivos se pueden perder bastantes imágenes de cara al entrenamiento.

No obstante un filtrado puede tener sus limites, tal vez puede no ser realizado por lo que habría que pensar otra metodología.

Base+Pre 1 (div=3)+Post1. En este experimento se hace uso del pre procesado para la representación por tiempo con valor div 3, y consenso general entre *frames* como post procesamiento con la red FC-base. El objetivo es ver como mejora con un post procesado correcto y como dicho consenso permite mitigar el impacto de *frames* coincidentes o minoritarios con mala etiqueta sobre la etiqueta final de un vídeo.

Como se aprecia de nuevo las medias aumentan y se ve como todos los datos indican una mejor predicción y clasificación. El motivo por el cual se usa Pre 1(div=3) es porque es la metodología que mejores resultados ha mostrado. Este experimento muestra que un consenso o post procesado correcto puede mitigar el impacto de las etiquetas erróneas sobre las imágenes de un movimiento que sean similares a las de los demás.

Sin embargo este método depende mucho de que la mayoría de las etiquetas sean correctas por lo que si no es el caso, o son razonablemente correctas, las métricas no mejorarán mucho o empeorarían. Lo cual indicaría que sería necesario un cambio en la metodología o representación.

Base+Pre 1 (div=3)+Post2. En este experimento se hace uso del pre procesado para la representación por tiempo con valor div 3, y consenso ponderado entre *frames* como post procesamiento con la red FC-base. El objetivo es ver como si damos mayor peso a los *frames* más significativos de cada movimiento se consigue un mejor resultado puesto que harán de la votación un proceso más robusto ante *frames* con resultado erróneo.

Sin embargo, con la votación ponderada pasa lo contrario que se estimaba. Los resultados salen peor a pesar de en teoría, ser un modelo que tendría en cuenta mejor la evolución del movimiento dando a cada *frame* la influencia en la votación que se merece. Ahora, esto se puede dar por dos motivos. La función no es óptima y no se adapta bien a la evolución del movimiento. Siendo que cada movimiento es diferente y cada individuo ejecuta el movimiento a una velocidad distinta. Otro motivo sería porque el etiquetado de base que se tiene no es el adecuado para que este procesamiento funcione de manera correcta.

Entender que un post procesado es difícil que se adapte bien a los movimientos. Pero aun así supone una mejora con métodos anteriores y presenta valores en cuanto a la agrupación por clases similares a Pre 1(div=2) pero peores que Pre 1(div=3) a pesar que se usa este método.

Conclusión. Como conclusión de este experimento hemos observado que es esencial un pre procesado que elimine esos *frames* coincidentes acentuando las diferencias entre movimientos. Se ha comprobado que entre todas las arquitecturas, la arquitectura con pre procesado Pre1(div = 3) y post procesado consenso general es la que mejor resultados muestra. Esto debido a que mitiga etiquetas incorrectas y elimina una gran cantidad de *frames* coincidentes.

5.2.2. Análisis de la representación *por número de eventos*

Los objetivos a conseguir son los mismo que los anteriormente descritos más la comprobación de ver si esta representación muestra una mejora frente a la representación anterior o es el caso contrario. La Tabla 5.4 muestra el resumen de los resultados.

Base En este experimento no se hace uso ni de pre procesados ni post procesados, se usan todos los *frames* extraídos directamente de los ficheros y estos procesados por la

Tabla 5.4: Resultados (Precisión y *recall*) de la clasificación con variaciones de la representación *por eventos*. Arquitectura de red *FC-base*.

	Precisión (avg. avg. per class)	*recall* (avg. avg. per class)	Precisión avg. per frame	*recall* avg. per frame
Base	0.34	0.27	0.31	0.26
Base+Pre2(n=20)	0.37	0.33	0.33	0.30
Base+Pre2(n=40)	0.41	0.35	0.33	0.29
Base+Pre2(n=40)+Post1	**0.60**	**0.53**	**0.46**	**0.41**
Base+Pre2(n=40)+Post2	0.26	0.24	0.28	0.27

red FC-base. El experimento sirve como una primera toma de contacto, entender como de bien funciona esta nueva tecnología y comparar con la representación por tiempo.

Los resultados son bajos pero presentan una mejora con respecto a la representación por tiempo. Esto indicaría que la representación por tiempo no es robusta como se dijo anteriormente. Esto se ve en una precisión más alta y *recalls* superiores.

Esto concluye que la representación por eventos es una mejor representación. Esto debido a que no muestra entre *frames* variaciones de números de eventos provocados por la evolución del movimiento en el tiempo y variaciones en la ejecución del movimiento por cada usuario al hacerlo de manera distinta, más violenta o incluso más corta o más larga la acción.

Base+Pre 2 (n=20) En este experimento se hace uso del pre procesado de eliminación de los 20 primeros *frames* y últimos de cada vídeo. Además no se hace uso de ningún post procesado y se entrena con la red FC-base. Con este experimento se busca una mejora con respecto al anterior. Confirmar que los datos mejoran tras un filtrado y si mejores que los resultados obtenidos por la representación por tiempo.

De nuevo los datos mejoran y se aprecia una mejor precisión y *recall*. No obstante se ve como los datos son peores que en la representación por tiempo. Esto es debido a que el filtrado no es óptimo. De nuevo no se busca un algoritmo perfecto sino indicaciones de como tratar esta nueva tecnología. Además el filtrado anterior era mucho más adecuado debido a que permite ajustar el filtrado mejor al solo aceptar los *frames* intermedios del movimiento entiendo por cada vídeo cual es el mínimo de eventos y el máximo grabado. En este caso eliminamos los *frames* iniciales y finales en una cierta medida. Sin embargo indicaría que una eliminación mayor puede mejorar los resultados así como hacer que pueda superar a la representación por tiempo.

Base+Pre 2 (n=40) En este experimento se hace uso del pre procesado de eliminación de los 40 primeros *frames* y últimos de cada vídeo. Además no se hace uso de ningún post procesado y se entrena con la red FC-base. Con este experimento se busca ver como un mejor filtrado hace mejorar los resultados.

Las medias mejoran debido a un mejor filtrado. Vemos como efectivamente en el experimento anterior que las medias fueran más bajas indicaba una falta de un filtrado más preciso y que la robustez de esta representación hace que los resultados sean mejores que los de la representación por tiempo. Esto debido a que hay los datos por *frame* son muy similares aun pudiéndose mejorar el filtrado aun más.

Sin embargo de nuevo, todo filtrado es mejorable pero también tiene un límite por lo que otra metodología podría ser mejor opción.

Base+Pre 2 (n=40)+Post1 En este experimento se hace uso del pre procesado de eliminación de los 40 primeros *frames* y últimos de cada vídeo. Además se hace uso de un consenso general como post procesado y se entrena con la red FC-base. Con este experimento se busca ver como se compara con la representación por tiempo para ver si definitivamente esta representación es mejor que la de por tiempo.

Los datos mejoran notablemente, esto debido a que el consenso hace que los *frames* minoritarios con mal etiquetado sean corregidos y mejoran contra la representación por tiempo. Definitivamente esta representación es más robusta y por tanto nos da mejores resultados. Resultados ya medianamente altos. Valores que empezarían a indicar que esta nueva tecnología pueda ser incorporada al *deep learning* y conseguir resultados razonables.

Base+Pre 2 (n=40)+Post2 En este experimento se hace uso del pre procesado de eliminación de los 40 primeros *frames* y últimos de cada vídeo. Además se hace uso de un consenso ponderado como post procesado y se entrena con la red FC-base. Con este experimento se busca ver como se compara con la representación por tiempo para ver si definitivamente esta representación es mejor que la de por tiempo y si el consenso ponderado es mejor o peor que un consenso general.

Los datos son bajos, lo cual indicaría que para esta representación realizar este post procesado no es lo adecuado. Puesto que empeora bastante con respecto a su contra parte. El motivo que esta representación no sigue la evolución temporal del movimiento. Los *frames* son agrupaciones de eventos pero no se tiene en cuenta la temporalidad, por lo que un consenso general más que uno que busca influir más sobre los *frames* de las fases intermedias del movimiento resulta una mejor opción.

Conclusión. De los experimentos anteriores se puede concluir que la representación por eventos es mejor que la de tiempo y que la información de la evolución temporal será imprescindible para conseguir una mejora frente a la metodología anterior puesto que un mejor filtrado solo va a conseguir una pequeña mejora sobre los resultados.

Por lo tanto, en los experimentos siguientes consideramos buscar introducir la evolución del movimiento que tan solo se puede realizar introduciendo el vídeo o una secuencias de *frames* que lo muestren. Además de que se usa la codificación por número de eventos al ser la mejor para construir los *frames* dentro del conjunto.

5.2.3. Predicción por *frames* vs por conjuntos de *frames*.

Este experimento evalúa las mejoras obtenidas utilizando conjuntos de *frames* como entrada al clasificador en lugar de clasificar cada *frame* de manera independiente (como se ha explicado en Sec. 4). Se han evaluado tres pequeñas variaciones, considerando conjuntos con distinto numero de *frames* (NF): 3, 5 o 7 *frames*.

La Tabla 5.5 muestra el resumen de los resultados de este experimento, incluyendo la media de precision y *recall* por clase y por frame. La precisión y el *recall* en esta última versión del sistema son ya cercanos al 60 %, mejorando en casi un 30 % los resultados obtenidos con arquitecturas existentes para clasificación de imagen (modelo Base). Además, también se aprecia que el número de *frames* por conjunto es un parámetro importante, que puede hacer que los resultados sean significativamente mejores.

El poder observar la progresión del movimiento es clave para poder realizar una correcta clasificación. Por el hecho de que se pueden apreciar los puntos característicos

Tabla 5.5: Resultados (Precisión y *recall*) de la clasificación con variaciones de la representación *por conjuntos de frames*.

	Precisión avg. per class	*recall* avg. per class	Precisión avg. per frame	*recall* avg. per frame
NF = 5	0.63	0.58	0.57	0.54
NF = 3	0.55	0.49	0.48	0.46
NF = 7	0.59	0.52	0.43	0.44

del movimiento en cada fase. Esto se puede comprobar mirando la matriz de confusión 5.3.

La matriz de confusión obtenida se acerca mas a lo ideal, es decir, bajas confusiones y gran número de etiquetado correcto (mostrando una diagonal más marcada en la matriz) . Esto muestra la importancia que tiene el poder estudiar y analizar la evolución temporal del movimiento. Además esto mostraría que aun puede mejorarse el pre-procesado y buscar técnicas que mostrarán mejor esa evolución temporal.

Como se ha mencionado, todos estos análisis se realizan con la configuración red FC-base (Cap. 4), para poder realizar los experimentos y sus variaciones en menor tiempo (debido a los requisitos de tiempo y hardware ser altos como para poder ejecutar sobre el sistema personal de trabajo arquitecturas complejas).

Se quiere sin embargo evaluar si una red más compleja haría que los resultados fueran mucho mejores. Para ello, se han repetido los experimentos con la red más compleja mencionada (Resnet50V2[19]), con la mejor configuración obtenida hasta ahora (Variación NF=5). Estos resultados se muestran en la Tabla 5.6.

Tabla 5.6: Resultados (Precisión y *recall*) utilizando la arquitectura más compleja (ResNet50V2[19]). Variación NF=5 de la representación *por conjuntos de frames*

Red Compleja	avg. avg. per class	avg. per *frame*
Precisión	0.74	0.68
recall	0.76	0.72

Con esta arquitectura de red más sofisticada y compleja vemos como los resultados son significativamente mejores. La precisión aumenta mucho colocándose casi en el 80 % además de tener un *recall* alto de media cerca del 70 %. La matriz de confusiones 5.4 se muestra mucho mejor, con aun más bajas confusiones no tan esparcidas entre las clases y el mayor número de clasificaciones asociado a la clase correcta.

Como se puede ver en la matriz de confusión 5.4, las clases girar y saltar siguen teniendo resultados bastante malos. Estas dos clases en si son muy parecidas se diferencian en puntos claves del movimiento que duran cuestión de milésimas de segundo. Lo cual hace extremadamente difícil su etiquetado correcto.

Conclusión. Con este último experimento aun se ha conseguido mejorar mas el sistema de base. Hemos verificado que las arquitecturas, sencillas o más complejas, para clasificación de imagen son un gran punto de partida pero ha sido necesario proponer una serie de adaptaciones para los datos de eventos para conseguir resultados mas robustos.

5.2.4. Validación con datos propios.

En este experimento se comprueban las ventajas que aportan las cámaras de eventos contra las tradicionales mostrando resultados obtenidos sobre escenarios donde lo tradicional no puede operar: baja luminosidad y alta velocidad. También queremos ver si podemos replicar los resultados obtenidos anteriormente con el otro *data set*.

Tabla 5.7: Resultados (Precisión y *recall*) de la clasificación con variaciones en luz y velocidad de ejecución del movimiento.

	Precisión avg. per class	*recall* avg. per class	Precisión avg. per frame	*recall* avg. per frame
Luz débil/velocidad normal	0.05	0.06	0.05	0.06
Luz débil/velocidad rápida	0.07	0.07	0.06	0.06
Luz normal/velocidad normal	0.71	0.40	0.63	0.30
Luz normal/velocidad rápida	0.61	0.39	0.20	0.26

Los resultados como se ven Tabla 5.7, en los casos de luz débil, en particular, son bajos. Algunos son algo positivos pero si se miran dichos porcentajes se pueden ver que los resultados en nada se parecen a los anteriores. Por lo tanto que se extraen de estos: Lo primero es entender el porque han fallado tanto. Las grabaciones, como se puede apreciar en la Fig. 5.5, no se parecen en nada a las usadas en el entrenamiento, mostrándose *frames* de mucha peor calidad donde no se ve al individuo y solo fragmentos del movimiento.

Esto provoca que ya desde un inicio la experimentación vaya a fallar. No obstante, recordar que el objetivo es ver si se pueden obtener resultados similares a los obtenidos anteriormente y si la cámara aun funciona en escenarios donde las tradicionales no pueden.

Para ello hay que fijarse en los casos de alta velocidad y iluminación débil. En un inicio, movimientos a velocidad alta, el movimiento dura unos 2 segundos, es capaz de captar como se ve en el anexo algunas clases con alta precisión y *recall*. Sin embargo falla en otras, pero también es normal puesto que son las peores clases extraídas del entrenamiento realizado con los datos anteriores.

El mayor problema se encuentra con la iluminación débil. No obstante, aclarar dos cuestiones. Es normal que los datos salgan peor, puesto que las grabaciones como se muestra aquí 5.5 son incluso peores que en los otros escenarios con iluminación normal. Además que en estos escenarios la cámara tradicional no podría operar puesto que es oscuridad casi absoluta. Por lo que el que muestre algún tipo de respuesta ya es todo un mérito.

Además de mostrar que es capaz de reconocer una clase, si con precisiones y *recalls* bajos pero aun decentes dadas las circunstancias. Por lo que esto demuestra que estas tecnologías son más capaces que las tradicionales en este tipo de escenarios. Lo que indicaría que con una mejora considerable de las grabaciones se pondrían obtener resultados mucho mejores.

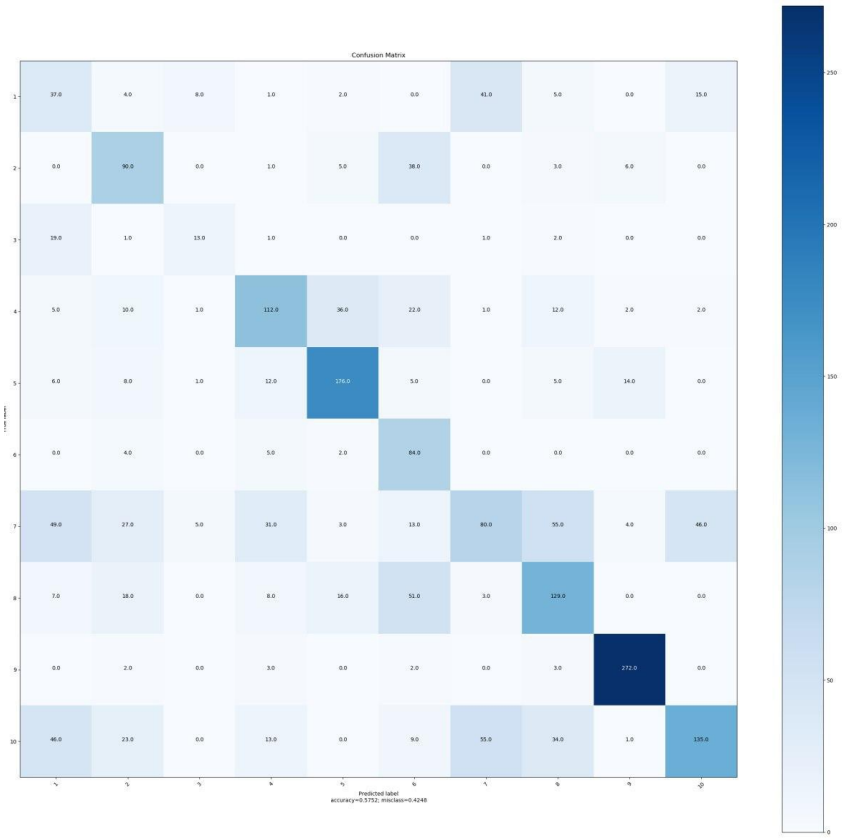


Figura 5.3: Matriz de confusión por cada clase del *data setNeuromorphic Benchmark* [15]. Configuración **n=5** y arquitectura de red FC-Base. Representación de eventos *por conjunto de frames*. En escala de azules se indica el número de etiquetas para cada una de las clases del movimiento. Cuanto más oscuro es el color azul mayor es el número de etiquetas y a la inversa, cuanto más claro menor es el número. Cada casilla por tanto simboliza el número de etiquetas de una acción respecto de otra (predicción vs instancia real), es decir la casilla C_i_j tiene las etiquetas correspondientes a la acción i frente a la j. El orden de las acciones es el siguiente de la casilla más a la izquierda a la casilla más a la derecha (mismo para filas que para columnas): Cruce de brazos, levantarse, dar una patada, coger algo del suelo, saltar, sentarse, lanzar, darse la vuelta, andar y saludar.

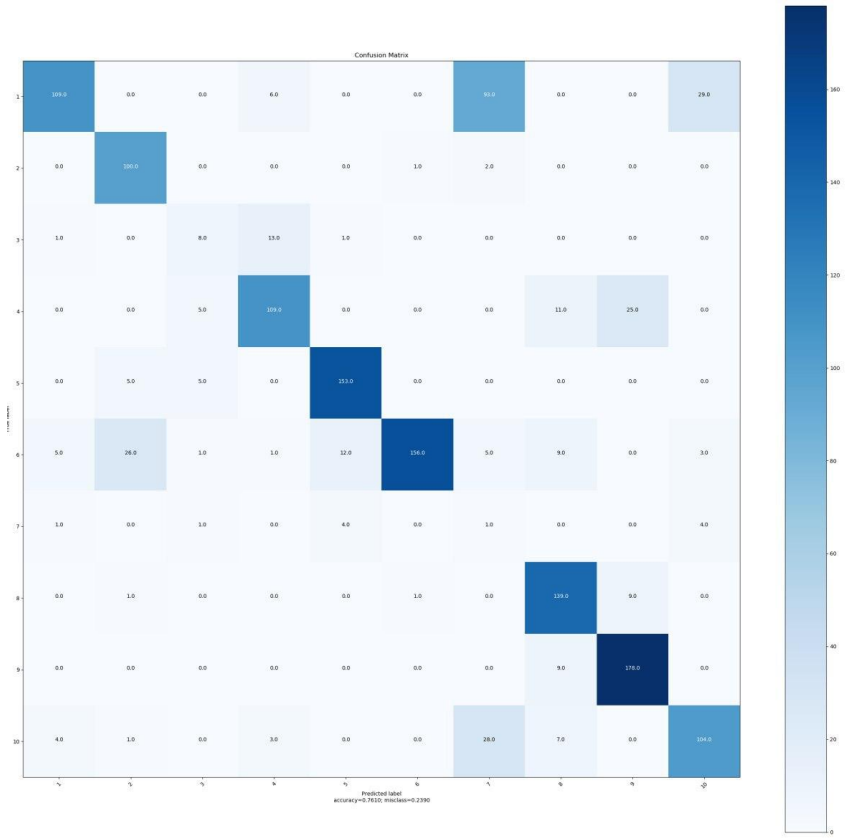
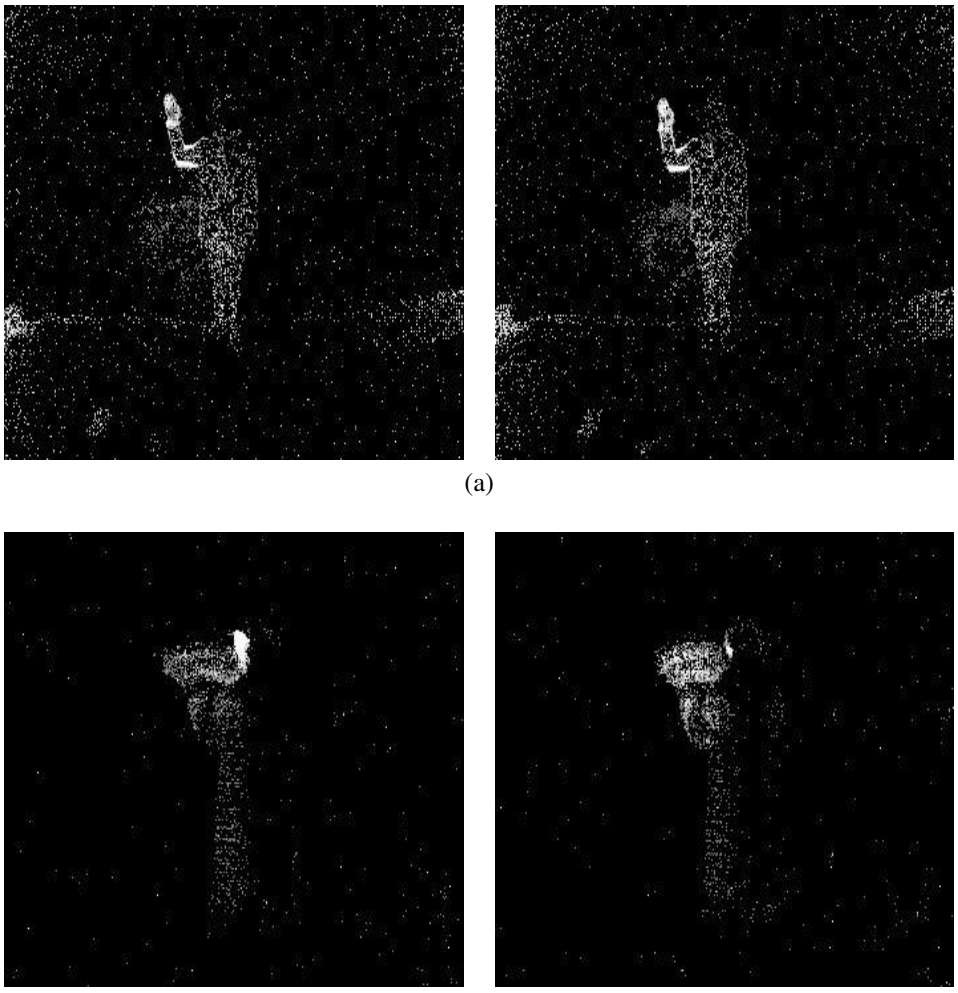


Figura 5.4: Matriz de confusión por cada clase del *data setNeuromorphic Benchmark* [15]. Configuración **n=5** y arquitectura de red ResNet50V2. Representación de eventos *por conjunto de frames*. En escala de azules se indica el número de etiquetas para cada una de las clases del movimiento. Cuanto más oscuro es el color azul mayor es el número de etiquetas y a la inversa, cuanto más claro menor es el número. Cada casilla por tanto simboliza el número de etiquetas de una acción respecto de otra (predicción vs instancia real), es decir la casilla C_i_j tiene las etiquetas correspondientes a la acción i frente a la j. El orden de las acciones es el siguiente de la casilla más a la izquierda a la casilla más a la derecha (mismo para filas que para columnas): Cruce de brazos, levantarse, dar una patada, coger algo del suelo, saltar, sentarse, lanzar, darse la vuelta, andar y saludar.



(a)

(b)

Figura 5.5: Comparativa de *frames* entre la misma acción en el dataset Neuromorphic Benchmark[15] (a) y el propio grabado (b). Los dos *frames* de cada set corresponden con el movimiento de saludar en ambos casos. Se puede apreciar como, pese a que las condiciones de luz y velocidad en el dataset propio son las mismas que en el dataset Neuromorphic Benchmark[15], las grabaciones son bastante diferentes.

Capítulo 6

Conclusión

6.1. Conclusiones Técnicas

En este trabajo se ha presentado un estudio sobre el uso de las cámaras de eventos en la tarea de reconocimiento de acciones. El objetivo principal era establecer si esta nueva tecnología podía ser compatible con los métodos actuales usados en visión por computador, y si tenían algún tipo de ventaja frente a las cámaras tradicionales.

Se han evaluado métodos estándares existentes que han conseguido resultados razonables adaptados de manera específica para trabajar con eventos.

Se ha visto como cada una de las representaciones muestran ventajas y desventajas estableciendo al final cual de ellas es la mejor opción para trabajar con eventos, así como proponer una nueva forma de representación de eventos que ha mostrado resultados incluso mejores que las anteriores representaciones convencionales.

Además se ha comprobado como los métodos propuestos tanto en las fases de pre y post procesado han mejorado los resultados sobre los métodos más estándares actuales en la literatura del reconocimiento de acciones sobre cámaras de eventos.

Finalmente se ha concluido que las cámaras de eventos poseen ventajas sobre las cámaras tradicionales. Estas pueden funcionar sobre escenarios donde los *frames* obtenidos por cámaras tradicionales no capturan información significativa.

Problemas encontrados. Los principales problemas encontrados han sido los siguientes:

- Imposibilidad de conseguir replicar las grabaciones obtenidas sobre los escenarios propuestos de baja luz. Aunque se ha podido grabar con esta cámara, en algunos escenarios, en particular los de baja luz, ha sido bastante difícil conseguir grabaciones de calidad. Esto debido a una falta de experiencia y conocimiento al detalle de las configuraciones de la cámara. Tras hablar con los propios fabricantes y realizar varios intentos se opto por quedarse con las mejores grabaciones y realizar los experimentos de todas formas. Pues se buscaba ver si aun estas cámaras en estos escenarios extremos podían mostrar resultados.

- Dificultad para realizar entrenamientos complejos y con grandes cantidades de datos sobre el sistema disponible. Con la reciente pandemia el acceso a los laboratorios fue imposible y realizar dichos entrenamientos en el sistema de trabajo personal no era suficiente. Los grandes costes temporales y las exigencias de

hardware (tarjeta gráfica de rendimiento) hacían de la tarea de entrenamiento un proceso arduo y lento. Por lo que debido a esto se decidió realizar entrenamientos con arquitecturas simples y rápidas, aunque aun así el entrenamiento seguía siendo largo, pero que aminoraban la carga y agilizaban el proceso. Al final el mejor sistema sería entrenado por la red más compleja para tener resultados sobre un sistema no básico.

- Búsqueda de literatura, experimentos similares, o *data sets* de calidad sobre los que basar esta investigación. No es mucha la información actual sobre esta nueva tecnología y los trabajos hasta la fecha son básicos y guían sobre formas de tratar y tener un primer contacto con las cámaras de eventos. Esto dificulto el encontrar un trabajo sobre el que apoyar este. También la falta de conjuntos de datos estándares y consensuados hizo que en una primera toma del trabajo se tuviera que probar dichos conjuntos de datos hasta dar con el adecuado. Esto supuso que el trabajo fuera realizado con cierta incertidumbre e intentando siempre conseguir el mejor resultado.

6.2. Conclusiones Personales

A parte del hecho que me gustan los campos de visión por computador, este es mi primer trabajo académico considerable, que involucra en cierto modo, un trabajo de investigación. Algo que buscaba para obtener más experiencia y ver cómo sería tomar un trabajo sobre una tecnología nueva. Mi opinión personal es que esta tecnología sirve en concreto de soporte y no de reemplazo a la tradicional. No podemos descartar cuestiones como el detalle que dan los *frames* tradicionales.

Sin embargo, si hay escenarios donde esta tecnología puede reemplazar a la anterior por varias cuestiones. La primera y más relevante de todas es que es independiente de la luminosidad. Segundo su poco gasto energético. Tercero su baja latencia. Por ultimo, algo que nunca se ha comentado, y es su poca intrusión. Estas cámaras captan eventos no imágenes tal cual las vemos. Esto provoca que la privacidad sea mucho más respetada.

En definitiva, personalmente, veo las cámaras de eventos como un medio para avanzar en los campos de visión por computador, una tecnología adaptable a diferentes ámbitos.

6.3. Trabajo Futuro

Tras este trabajo aun son muchas las cuestiones sobre las que indagar y profundizar. Varias preguntas aun quedan abiertas y en futuros trabajos se podrían tratar muchos otros aspectos para diseñar o adaptar técnicas existentes de *deep learning* para avanzar en esta novedosa tecnología y sus aplicaciones.

En concreto para el reconocimiento de acciones, algunas de ellas son:

- Buscar métodos de pre/post procesado que mejores que consigan que los *frames* más significativos tengan más peso.

- Buscar metodologías que abaraten el coste temporal sobre arquitecturas complejas.

- Mejora de las representaciones para que sean capaces de capturar cambios pequeños pero muy significativos para distinguir acciones muy similares, o consigan ser mas invariantes al escenario alrededor de la acción.
- Aplicación de otras arquitecturas de *deep learning* que tienen en cuenta directamente la información temporal.

Bibliografía

[1] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Jörg Conradt, Kostas Daniilidis, et al. Event-based vision: A survey. *arXiv preprint arXiv:1904.08405*, 2019.

[2] Lin Wang, Yo-Sung Ho, Kuk-Jin Yoon, et al. Event-based high dynamic range image and very high frame rate video generation using conditional generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10081–10090, 2019.

[3] Guillermo Gallego and Davide Scaramuzza. Accurate angular velocity estimation with an event camera. *IEEE Robotics and Automation Letters*, 2(2):632–639, 2017.

[4] Guillermo Gallego, Jon EA Lund, Elias Mueggler, Henri Rebecq, Tobi Delbruck, and Davide Scaramuzza. Event-based, 6-dof camera tracking from photometric depth maps. *IEEE transactions on pattern analysis and machine intelligence*, 40(10):2402–2412, 2017.

[5] Henri Rebecq, Timo Horstschäfer, Guillermo Gallego, and Davide Scaramuzza. Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real time. *IEEE Robotics and Automation Letters*, 2(2):593–600, 2016.

[6] Timo Stoffregen, Guillermo Gallego, Tom Drummond, Lindsay Kleeman, and Davide Scaramuzza. Event-based motion segmentation by motion compensation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7244–7253, 2019.

[7] Davide Falanga, Kevin Kleber, and Davide Scaramuzza. Dynamic obstacle avoidance for quadrotors with event cameras. *Science Robotics*, 5(40), 2020.

[8] 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. *IEEE Robotics and Automation Letters*, 3(2):994–1001, 2018.

[9] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *CVPR 2011*, pages 3169–3176. IEEE, 2011.

[10] Ivan Laptev. On space-time interest points. *International journal of computer vision*, 64(2-3):107–123, 2005.

[11] Piotr Dollár, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. Behavior recognition via sparse spatio-temporal features. In *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 65–72. IEEE, 2005.

[12] Xiaojiang Peng, Limin Wang, Xingxing Wang, and Yu Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *Computer Vision and Image Understanding*, 150:109–125, 2016.

[13] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[14] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[15] Shu Miao, Guang Chen, Xiangyu Ning, Yang Zi, Kejia Ren, Zhenshan Bing, and Alois C Knoll. Neuromorphic benchmark datasets for pedestrian detection, action recognition, and fall detection. *Frontiers in neurorobotics*, 13:38, 2019.

[16] 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 Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[17] Shane Harrigan, Sonya Coleman, Dermot Kerr, Y Pratheepan, Fang Zheng, and Chengdong Wu. Neuromorphic event-based action recognition. In *Irish Machine Vision and Image Processing Conference*, pages 218–221. Irish Pattern Recognition and Classification Society, 2019.

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

[19] TensorFlow. Resnet50v2. `https://www.tensorflow.org/api_docs/python/tf/keras/applications/ResNet50V2`.

[20] 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.

[21] iniVation. Aedat file formats. `https://inivation.github.io/inivation-docs/Software%20user%20guides/AEDAT_file_formats.html`.

[22] Google. Flatbuffers: Flatbuffers. `https://google.github.io/flatbuffers/`.

APÉNDICES- Reconocimiento de gestos por Cámara de eventos con técnicas de *Deep Learning*

Apéndice A

Formato AEDAT

AEDAT (Address Event DATA) [21] es un formato para el almacenamiento de información de cámaras de eventos y el *timestamp* en el cual son capturados. Estos ficheros son generados por el software basado en eventos, DV y jAER (ambas plataformas para la visualización de datos de eventos) de la compañía *iniVation*. Siendo este el usado en este trabajo debido que el modelo de la cámara usada en los experimentos, prestada por parte del laboratorio, *DAVIS346* pertenece a esta compañía, además de ser el principal formato usado en la mayoría de los investigaciones hasta la fecha.

Durante los años, este formato ha ido evolucionado para satisfacer las necesidades que han ido surgiendo en el desarrollo de este tipo de tecnología por eventos. En la actualidad se hace uso del aedat versión 4.0.

Esta versión hace uso de los llamados Google *Flatbuffers* [22] para serializar los datos de forma conveniente y eficiente, con *path* para poder extender estructuras ya existentes de datos de forma sencilla e introducir nuevas. Estos también permiten fácil y rápido soporte a otros lenguajes como Python o Java, auto generando los ficheros necesarios para estos lenguajes. Las características de estos FlatBuffers son las siguientes:

- Acceso a datos serializados sin la necesidad de parseo o desempaquetar. Lo que separa a estos es que representan una estructura jerárquica plana de datos binarios que puede ser accedida directamente sin necesidad de parseo o de desempaquetar estos. Mientras que permiten la evolución de la estructura de datos.
- Eficiencia en memoria y velocidad. La única memoria necesaria para poder acceder a los datos es la del *buffer*. No requiere reserva de memoria. La velocidad de acceso esta cerca de la velocidad de acceso a estructuras en crudo con una única indirección para poder permitir la evolución de la estructura y campos opcionales.
- Flexible. Los campos opcionales permiten elegir que datos escribir y cuales no, así como diseñar la estructura deseada permitiendo la evolución de los datos y la creación de nuevas estructuras.
- Pequeña huella de código. Pequeñas cantidades de código generado, y un simple encabezamiento como mínima dependencia, que es muy fácil de integrar.
- Tipado fuerte. Los errores ocurren en tiempo de compilado en vez de en tiempo de ejecución.

Todos los *Flatbuffers* tienen un tamaño prefijado, significando que los primeros cuatro bytes representan una codificación de 32 bits en *little-endian* del tamaño actual de este.

Todos los *timestamps* dentro de los datos son enteros de 64 bits, y representan Unix time en microsegundos.

Los paquetes de datos consisten de un encabezamiento que identifica el *stream* y el tamaño del contenido a continuación.

El contenido puede estar comprimido o no. Si esta comprimido entonces todo el bloque de datos debe de ser descomprimido por los descompresores LZ4 o ZSTD's. El contenido tras la des comprensión es un *Flatbuffer* de tamaño prefijado.

Después del apartado de datos se encuentran o se pueden encontrar uno o más *FlatBuffers*, conocido como *FileDataTable*, que contiene la información de todos los paquetes de datos escritos anteriormente.

Apéndice B

Resultados adicionales de los experimentos

Este apéndice detalles adicionales de los experimentos realizados para validar los sistemas propuestos. En particular se incluyen:

- Todas las tablas con las medias de precisión y ***Recall*** en detalle para cada uno de los movimientos en los data sets usados.
  - La Tabla B.1 muestra el experimento Base para la representación *por tiempo*.
  - La Tabla B.2 muestra el experimento Base+Pre1(div=2) para la representación *por tiempo*.
  - La Tabla B.3 muestra el experimento Base+Pre1(div=3) para la representación *por tiempo*.
  - La Tabla B.4 muestra el experimento Base+Pre1(div=3)+Post1 para la representación *por tiempo*.
  - La Tabla B.5 muestra el experimento Base+Pre1(div=3)+Post2 para la representación *por tiempo*.
  - La Tabla B.6 muestra el experimento Base para la representación *por eventos*.
  - La Tabla B.7 muestra el experimento Base+Pre2(n=20) para la representación *por eventos*.
  - La Tabla B.8 muestra el experimento Base+Pre2(n=40) para la representación *por eventos*.
  - La Tabla B.9 muestra el experimento Base+Pre2(n=40)+Post1 para la representación *por eventos*.
  - La Tabla B.10 muestra el experimento Base+Pre2(n=40)+Post2 para la representación *por eventos*.
  - La Tabla B.12 muestra el experimento variación n=3 para la representación *por conjunto de eventos*.
  - La Tabla B.11 muestra el experimento variación n=5 para la representación *por conjunto de eventos*.

- La Tabla B.13 muestra el experimento variación n=7 para la representación *por conjunto de eventos*.
- La Tabla B.14 muestra el experimento variación n=5 con red Resnet50V2 para la representación *por conjunto de eventos*.
- La Tabla B.15 muestra el experimento luz débil y velocidad normal para la representación *por conjunto de eventos*.
- La Tabla B.16 muestra el experimento luz débil y velocidad rápida para la representación *por conjunto de eventos*.
- La Tabla B.17 muestra el experimento luz normal y velocidad normal para la representación *por conjunto de eventos*.
- La Tabla B.18 muestra el experimento luz normal y velocidad rápida para la representación *por conjunto de eventos*.

▪ Las matrices de confusión para todos los experimentos realizados.

- Las Figura B.1 se corresponde con la Tabla B.1 y su correspondiente experimento.
- Las Figura B.2 se corresponde con la Tabla B.2 y su correspondiente experimento.
- Las Figura B.3 se corresponde con la Tabla B.3 y su correspondiente experimento.
- Las Figura B.4 se corresponde con la Tabla B.4 y su correspondiente experimento.
- Las Figura B.5 se corresponde con la Tabla B.5 y su correspondiente experimento.
- Las Figura B.6 se corresponde con la Tabla B.6 y su correspondiente experimento.
- Las Figura B.7 se corresponde con la Tabla B.7 y su correspondiente experimento.
- Las Figura B.8 se corresponde con la Tabla B.8 y su correspondiente experimento.
- Las Figura B.9 se corresponde con la Tabla B.9 y su correspondiente experimento.
- Las Figura B.10 se corresponde con la Tabla B.10 y su correspondiente experimento.
- Las Figura B.12 se corresponde con la Tabla B.12 y su correspondiente experimento.
- Las Figura B.11 se corresponde con la Tabla B.11 y su correspondiente experimento.
- Las Figura B.13 se corresponde con la Tabla B.13 y su correspondiente experimento.
- Las Figura B.14 se corresponde con la Tabla B.14 y su correspondiente experimento.

Aclarar que todas las imágenes muestran lo siguiente: En escala de azules se indica el número de etiquetas para cada una de las clases del movimiento. Cuanto más oscuro es el color azul mayor es el número de etiquetas y a la inversa, cuanto más claro menor es el número. Cada casilla por tanto simboliza el número de etiquetas de una acción respecto de otra (predicción vs instancia real), es decir la casilla C_i_j tiene las etiquetas correspondientes a la acción i frente a la j. El orden de las acciones es el siguiente de la casilla más a la izquierda a la casilla más a la derecha (mismo para filas que para columnas): Cruce de brazos, levantarse, dar una patada, coger algo del suelo, saltar, sentarse, lanzar, darse la vuelta, andar y saludar.

Tabla B.1: Precisión y ***Recall*** por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **Base** y arquitectura de red FC-Base. Representación de eventos *por tiempo*.

Clase	**Precision**	***Recall***
Cruzar brazos	0.22	0.08
Levantarse	0.44	0.20
Saltar	0.04	0.27
Patada	0.18	0.25
Coger del suelo	0.29	0.16
Sentarse	0.38	0.27
Lanzar	0.16	0.26
Girar	0.26	0.27
Andar	0.35	0.39
Saludar	0.18	0.13
Media	0.25	0.23

Tabla B.2: Precisión y ***Recall*** por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **Base+Pre1(div=2)** y arquitectura de red FC-Base. Representación de eventos *por tiempo*.

Clases	Precisión	***Recall***
Cruzar brazos	0.34	0.21
Levantarse	0.38	0.11
Saltar	0.10	0.04
Patada	0.30	0.41
Coger del suelo	0.51	0.64
Sentarse	0.33	0.09
Lanzar	0.29	0.30
Girar	0.42	0.50
Andar	0.92	0.64
Saludar	0.52	0.76
Media	0.41	0.37

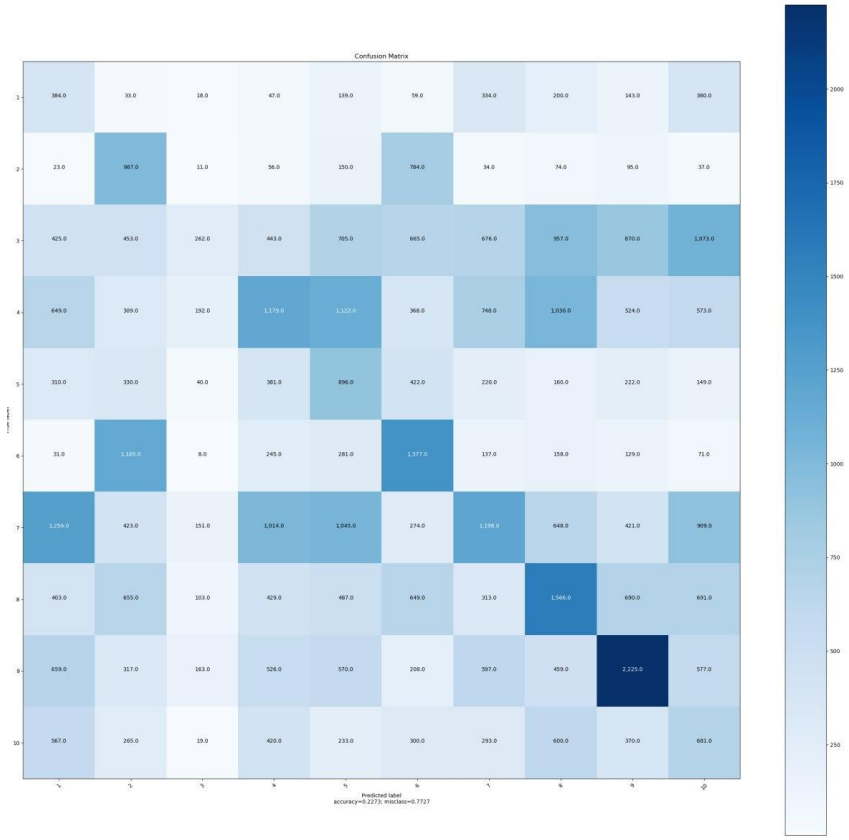


Figura B.1: Matriz de confusión por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **Base** y arquitectura de red FC-Base. Representación de eventos *por tiempo*.

Tabla B.3: Precisión y ***Recall*** por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **Base+Pre1(div=3)** y arquitectura de red FC-Base. Representación de eventos *por tiempo*.

Clases	Precisión	***Recall***
Cruzar brazos	0.36	0.24
Levantarse	0.44	0.20
Saltar	0.00	0.00
Patada	0.32	0.38
Coger del suelo	0.50	0.57
Sentarse	0.27	0.08
Lanzar	0.34	0.24
Girar	0.46	0.63
Andar	0.88	0.69
Saludar	0.62	0.89
Media	0.42	0.39

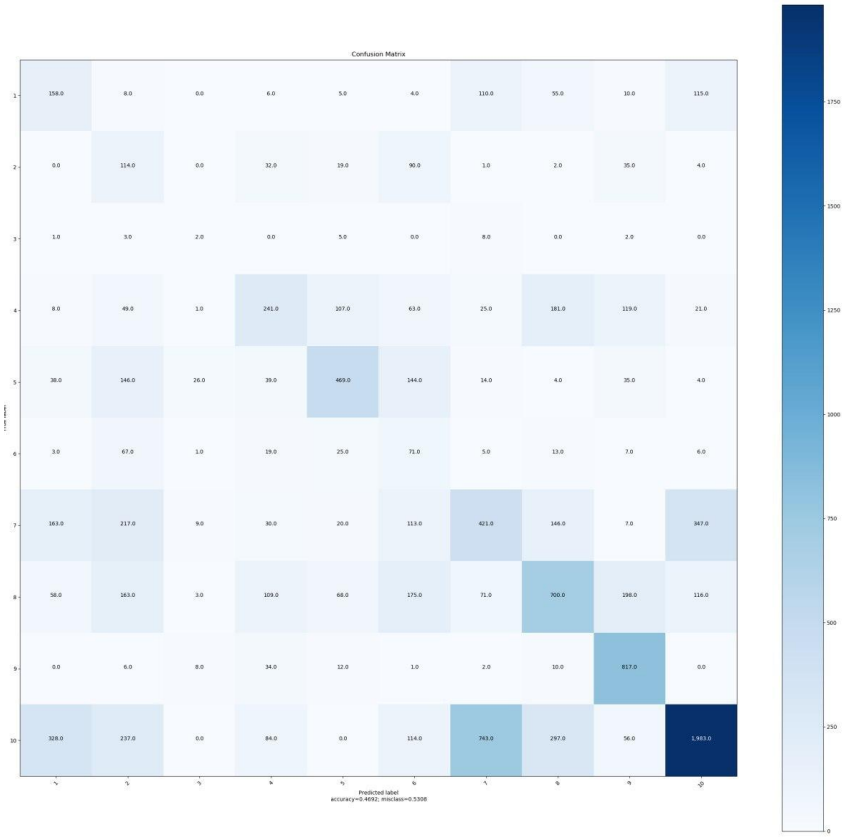


Figura B.2: Matriz de confusión por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **Base+Pre1(div=2)** y arquitectura de red FC-Base. Representación de eventos *por tiempo*.

Tabla B.4: Precisión y ***Recall*** por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **Base+Pre1(div=3)+Post1** y arquitectura de red FC-Base. Representación de eventos *por tiempo*.

Clases	Precisión	***Recall***
Cruzar brazos	0.41	0.20
Levantarse	0.58	0.23
Saltar	0.00	0.00
Patada	0.57	0.46
Coger del suelo	0.58	0.64
Sentarse	0.05	0.00
Lanzar	0.34	0.19
Girar	0.53	0.82
Andar	0.96	1.00
Saludar	0.62	0.94
Media	0.46	0.45

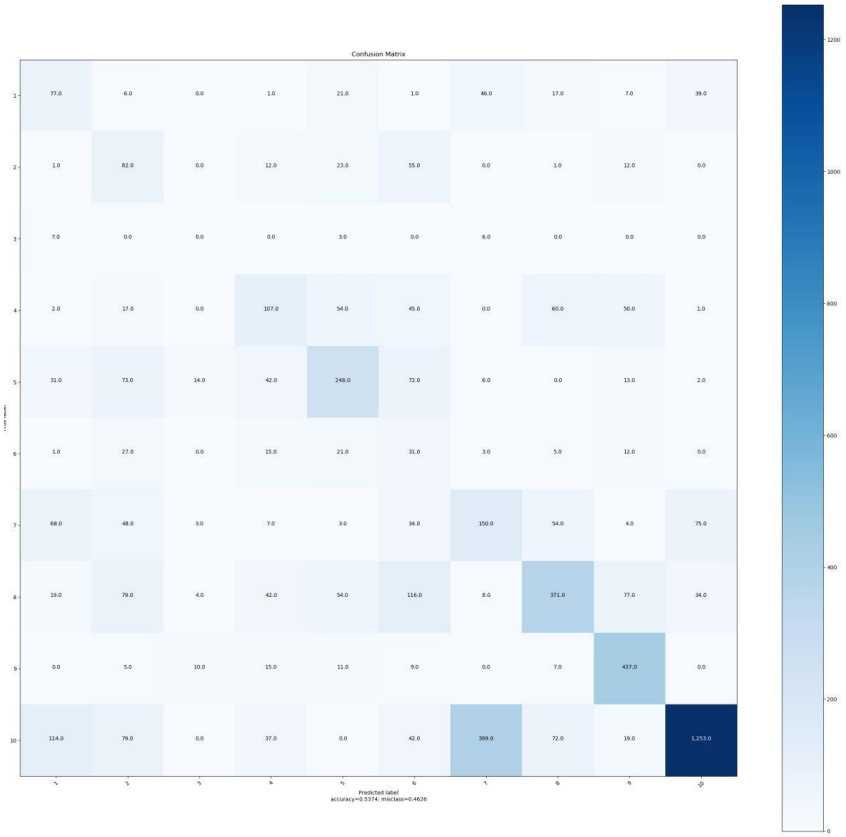


Figura B.3: Matriz de confusión por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **Base+Pre1(div=3)** y arquitectura de red FC-Base. Representación de eventos *por tiempo*.

Tabla B.5: Precisión y ***Recall*** por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **Base+Pre1(div=3)+Post2** y arquitectura de red FC-Base. Representación de eventos *por tiempo*.

Clases	Precisión	***Recall***
Cruzar brazos	0.50	0.51
Levantarse	0.33	0.08
Saltar	0.00	0.00
Patada	0.30	0.22
Coger del suelo	0.30	0.18
Sentarse	0.38	0.14
Lanzar	0.30	0.23
Girar	0.21	0.34
Andar	0.84	0.31
Saludar	0.53	0.91
Media	0.37	0.29

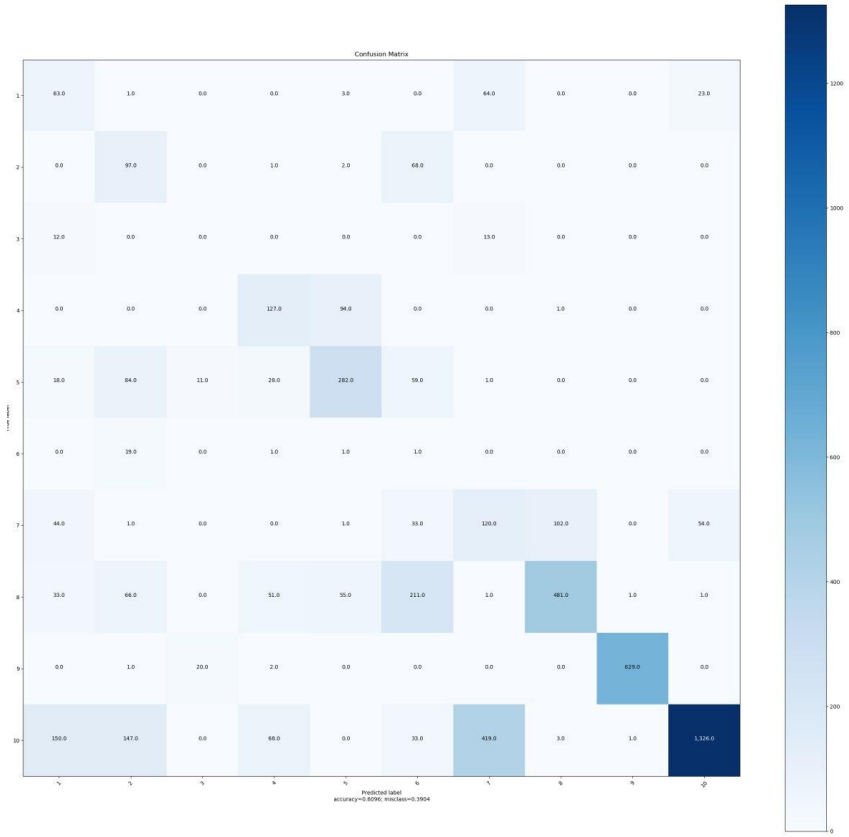


Figura B.4: Matriz de confusión por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **Base+Pre1(div=3)+Post1** y arquitectura de red FC-Base. Representación de eventos *por tiempo*.

Tabla B.6: Precisión y ***Recall*** por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **Base** y arquitectura de red FC-Base. Representación de eventos *por eventos*.

Clases	Precisión	***Recall***
Cruzar brazos	0.27	0.12
Levantarse	0.55	0.21
Saltar	0.04	0.36
Patada	0.20	0.32
Coger del suelo	0.30	0.29
Sentarse	0.38	0.26
Lanzar	0.18	0.16
Girar	0.35	0.21
Andar	0.59	0.54
Saludar	0.20	0.14
Media	0.31	0.26

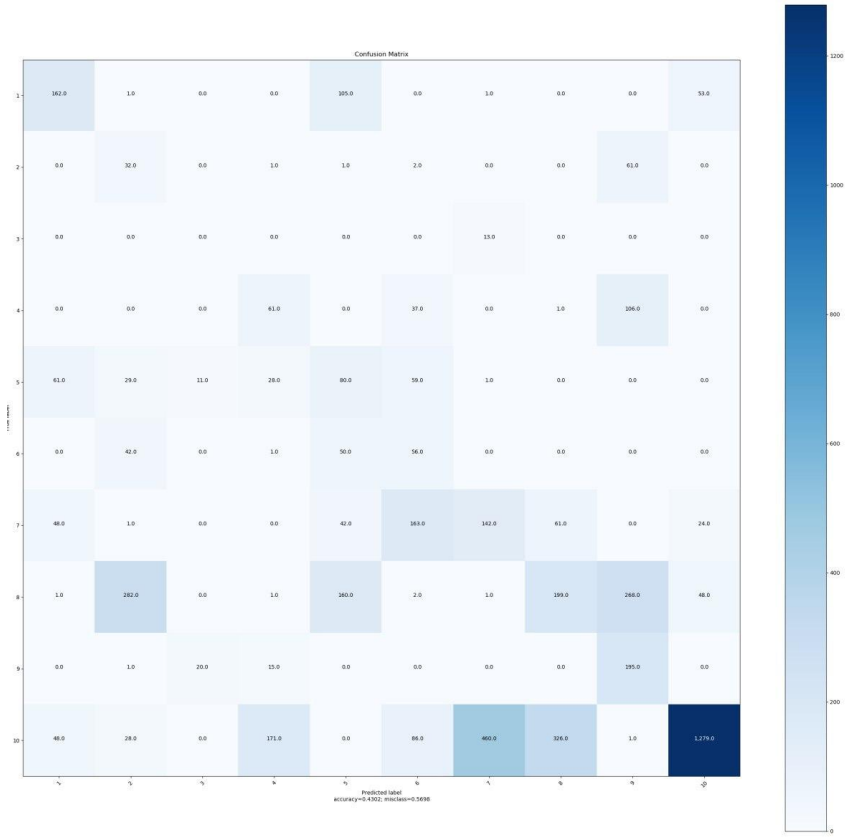


Figura B.5: Matriz de confusión por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **Base+Pre1(div=3)+Post2** y arquitectura de red FC-Base. Representación de eventos *por tiempo*.

Tabla B.7: Precisión y ***Recall*** por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **Base+Pre2(n=20)** y arquitectura de red FC-Base. Representación de eventos *por eventos*.

Clases	Precisión	***Recall***
Cruzar brazos	0.28	0.23
Levantarse	0.50	0.25
Saltar	0.05	0.20
Patada	0.24	0.36
Coger del suelo	0.31	0.41
Sentarse	0.43	0.21
Lanzar	0.19	0.20
Girar	0.38	0.33
Andar	0.61	0.55
Saludar	0.25	0.19
Media	0.33	0.30

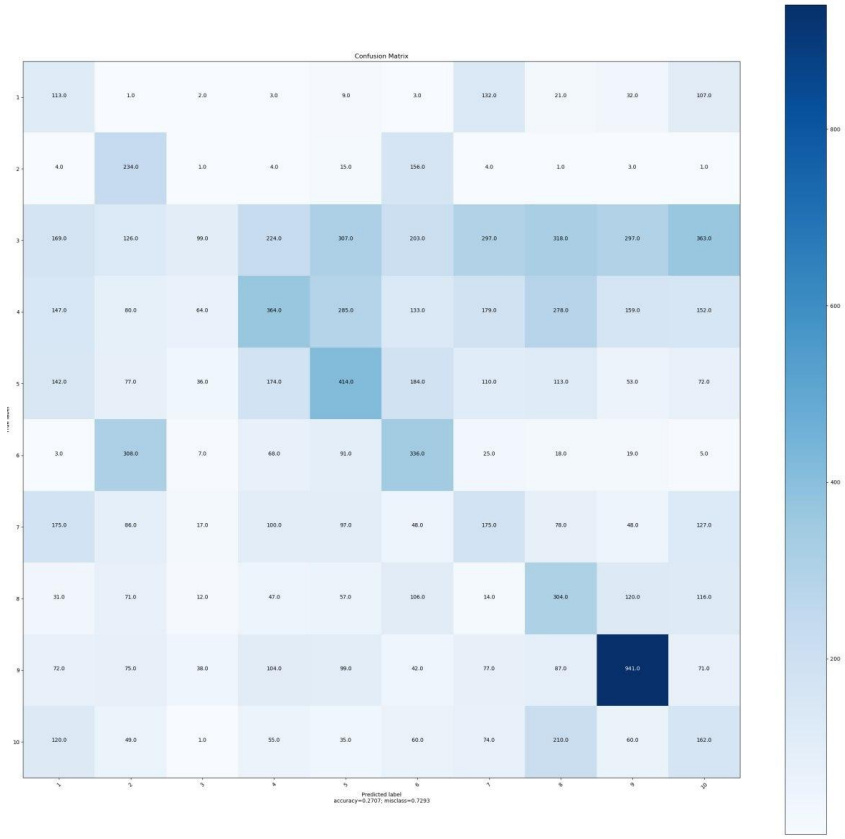


Figura B.6: Matriz de confusión por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **Base** y arquitectura de red FC-Base. Representación de eventos *por eventos*.

Tabla B.8: Precisión y ***Recall*** por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **Base+Pre2(n=40)** y arquitectura de red FC-Base. Representación de eventos *por eventos*.

Clases	Precisión	***Recall***
Cruzar brazos	0.14	0.17
Levantarse	0.62	0.49
Saltar	0.02	0.09
Patada	0.18	0.17
Coger del suelo	0.36	0.47
Sentarse	0.60	0.24
Lanzar	0.16	0.29
Girar	0.45	0.30
Andar	0.59	0.59
Saludar	0.21	0.09
Media	0.33	0.29

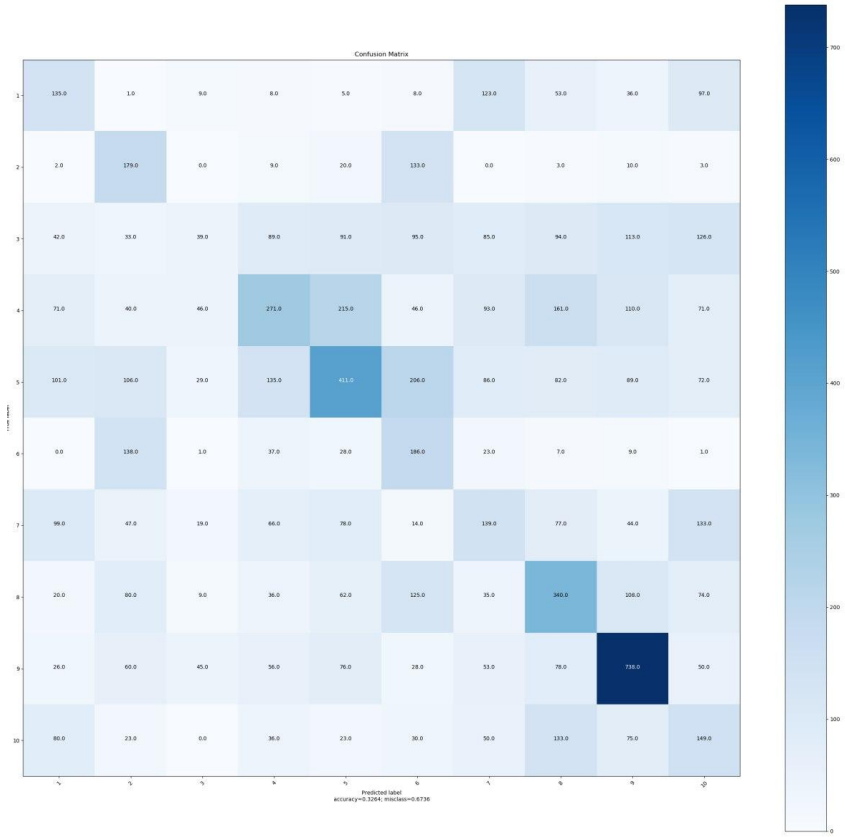


Figura B.7: Matriz de confusión por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **Base+Pre2(n=20)** y arquitectura de red FC-Base. Representación de eventos *por eventos*.

Tabla B.9: Precisión y ***Recall*** por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **Base+Pre2(n=40)+Post1** y arquitectura de red FC-Base. Representación de eventos *por eventos*.

Clases	Precisión	***Recall***
Cruzar brazos	0.22	0.24
Levantarse	0.72	0.60
Saltar	0.00	0.00
Patada	0.22	0.10
Coger del suelo	0.43	0.89
Sentarse	0.90	0.28
Lanzar	0.27	0.51
Girar	0.98	0.55
Andar	0.87	0.89
Saludar	0.02	0.01
Media	0.46	0.41

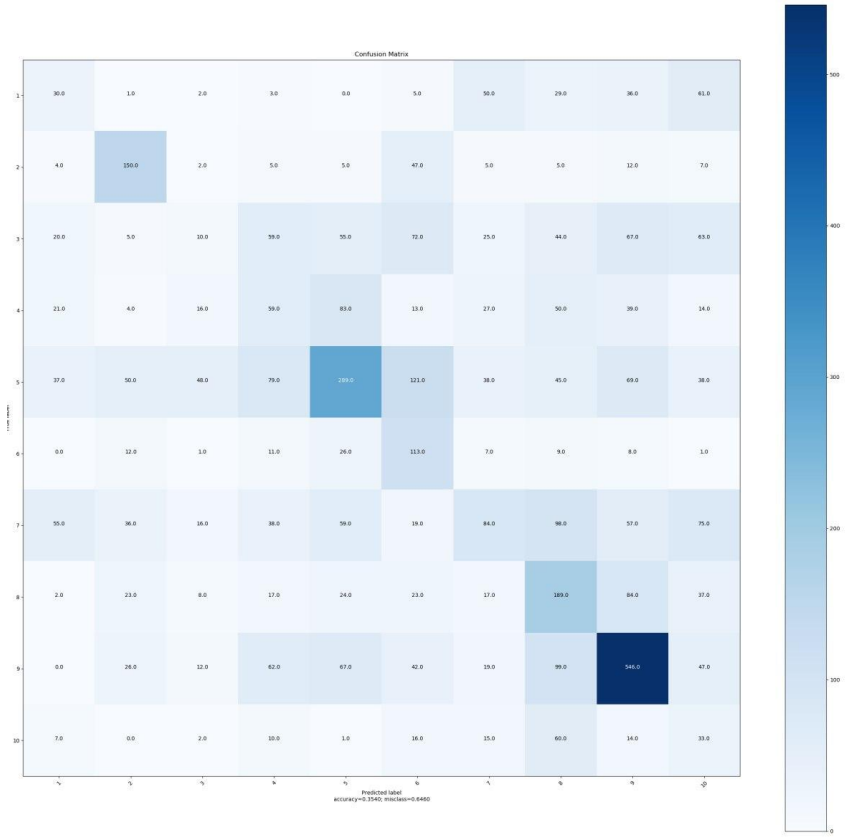


Figura B.8: Matriz de confusión por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **Base+Pre2(n=40)** y arquitectura de red FC-Base. Representación de eventos *por eventos*.

Tabla B.10: Precisión y ***Recall*** por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **Base+Pre2(n=40)+Post2** y arquitectura de red FC-Base. Representación de eventos *por eventos*.

Clases	Precisión	***Recall***
Cruzar brazos	0.18	0.20
Levantarse	0.70	0.60
Saltar	0.00	0.00
Patada	0.24	0.43
Coger del suelo	0.35	0.38
Sentarse	0.84	0.42
Lanzar	0.16	0.49
Girar	0.02	0.01
Andar	0.04	0.01
Saludar	0.23	0.15
Media	0.28	0.27

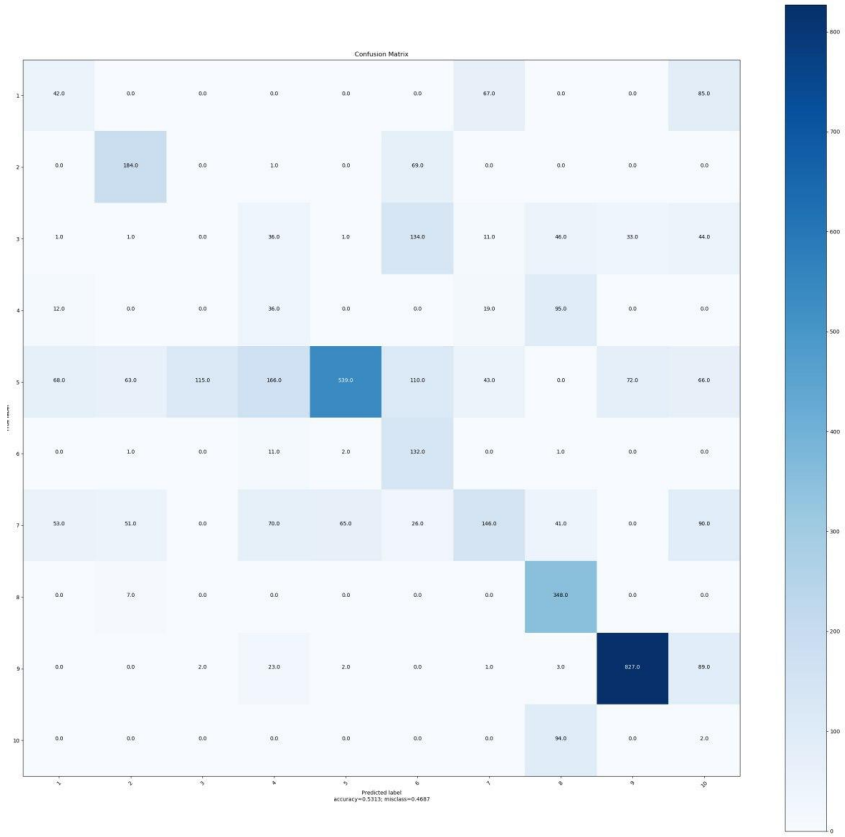


Figura B.9: Matriz de confusión por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **Base+Pre2(n=40)+Post1** y arquitectura de red FC-Base. Representación de eventos *por eventos*.

Tabla B.11: Precisión y ***Recall*** por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **n=5** y arquitectura de red FC-Base. Representación de eventos *por conjunto de frames*.

Clases	Precisión	***Recall***
Cruzar brazos	0.33	0.22
Levantarse	0.63	0.48
Saltar	0.35	0.46
Patada	0.55	0.60
Coger del suelo	0.78	0.73
Sentarse	0.88	0.38
Lanzar	0.26	0.44
Girar	0.56	0.52
Andar	0.96	0.91
Saludar	0.43	0.68
Media	0.57	0.54

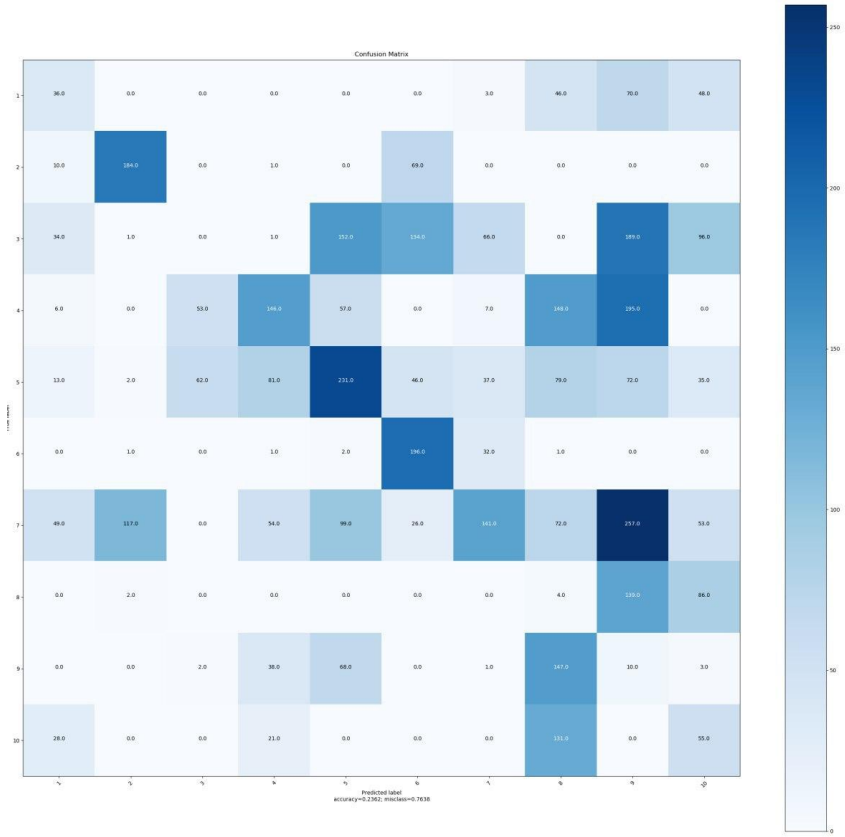


Figura B.10: Matriz de confusión por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **Base+Pre2(n=40)+Post2** y arquitectura de red FC-Base. Representación de eventos *por eventos*.

Tabla B.12: Precisión y ***Recall*** por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **n=3** y arquitectura de red FC-Base. Representación de eventos *por conjunto de frames*.

Clases	Precisión	***Recall***
Cruzar brazos	0.17	0.22
Levantarse	0.57	0.59
Saltar	0.11	0.38
Patada	0.45	0.52
Coger del suelo	0.64	0.47
Sentarse	0.84	0.37
Lanzar	0.29	0.11
Girar	0.56	0.43
Andar	0.92	0.93
Saludar	0.57	0.54
Media	0.48	0.46

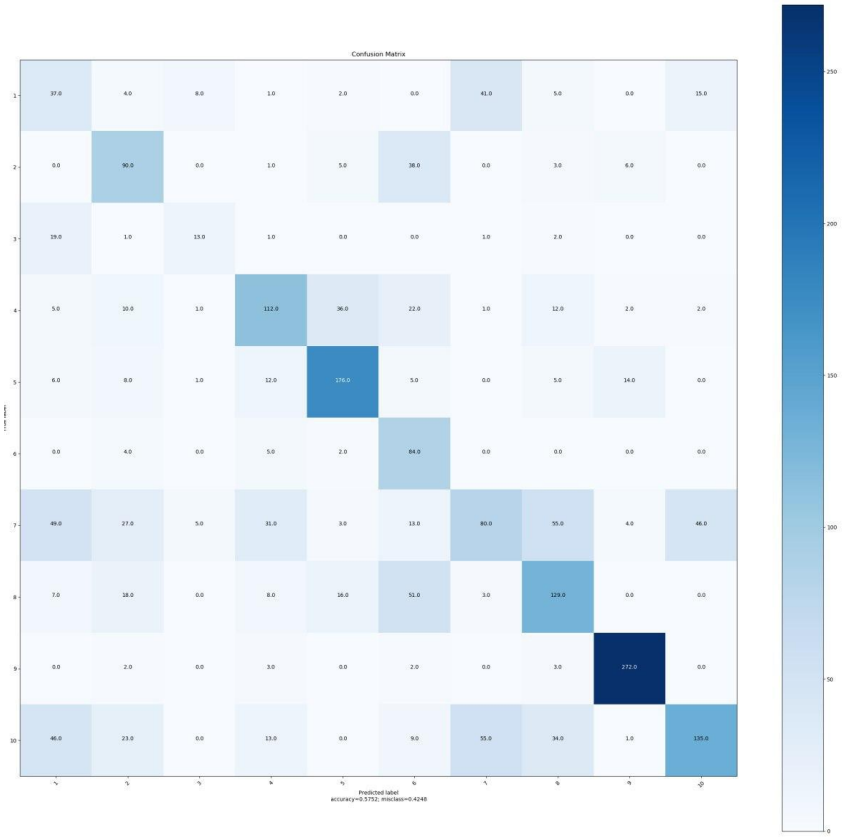


Figura B.11: Matriz de confusión por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **n=5** y arquitectura de red FC-Base. Representación de eventos *por conjunto de frames*.

Tabla B.13: Precisión y ***Recall*** por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **n=7** y arquitectura de red FC-Base. Representación de eventos *por conjunto de frames*.

Clases	Precisión	***Recall***
Cruzar brazos	0.24	0.14
Levantarse	1.00	0.39
Saltar	0.22	0.60
Patada	0.29	0.57
Coger del suelo	0.85	0.54
Sentarse	0.74	0.31
Lanzar	0.13	0.05
Girar	0.48	0.67
Andar	1.00	0.31
Saludar	0.26	0.79
Media	0.43	0.44

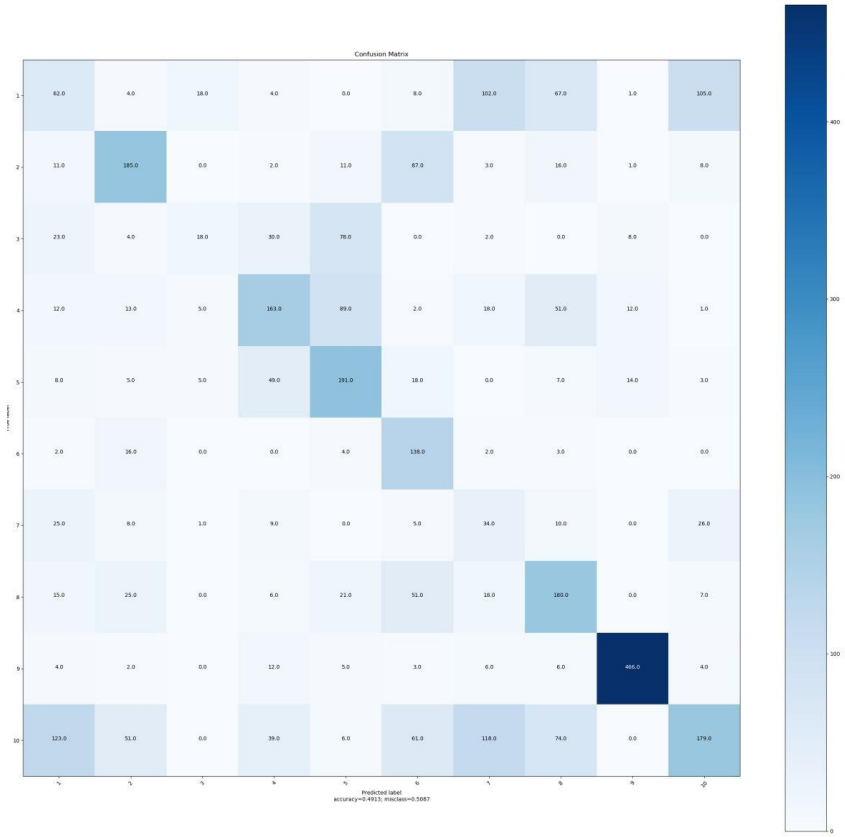


Figura B.12: Matriz de confusión por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **n=3** y arquitectura de red FC-Base. Representación de eventos *por conjunto de frames*.

Tabla B.14: Precisión y ***Recall*** por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **n=5** y arquitectura de red Red ResNet50V2. Representación de eventos *por conjunto de frames*.

Clases	Precisión	***Recall***
Cruzar brazos	0.46	0.91
Levantarse	0.97	0.75
Saltar	0.35	0.40
Patada	0.73	0.83
Coger del suelo	0.94	0.90
Sentarse	0.72	0.91
Lanzar	0.09	0.01
Girar	0.93	0.79
Andar	0.95	0.84
Saludar	0.71	0.74
Media	0.68	0.72

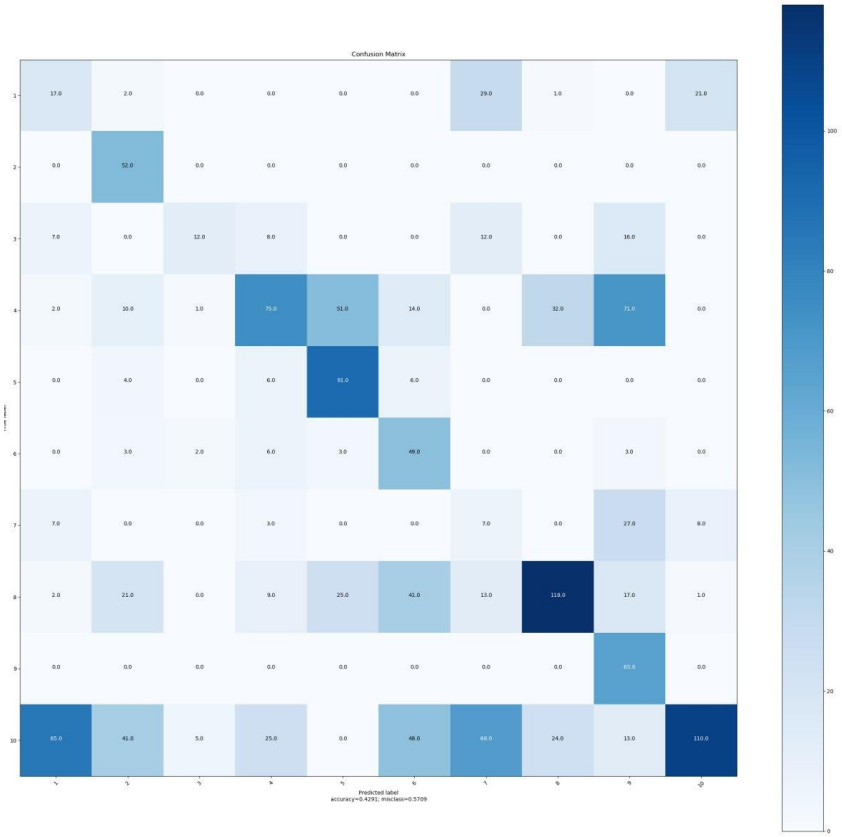


Figura B.13: Matriz de confusión por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **n=7** y arquitectura de red FC-Base. Representación de eventos *por tiempo*.

Tabla B.15: Precisión y ***Recall*** por cada clase del dataset propio. Configuración **n=5** y arquitectura de red ResNet50V2. Representación de eventos *por conjunto de frames*.

Clases	Precisión	***Recall***
Cruzar brazos	0.00	0.00
Levantarse	0.00	0.00
Coger del suelo	0.00	0.00
Andar	0.00	0.00
Saludar	0.26	0.30
Media	0.05	0.06

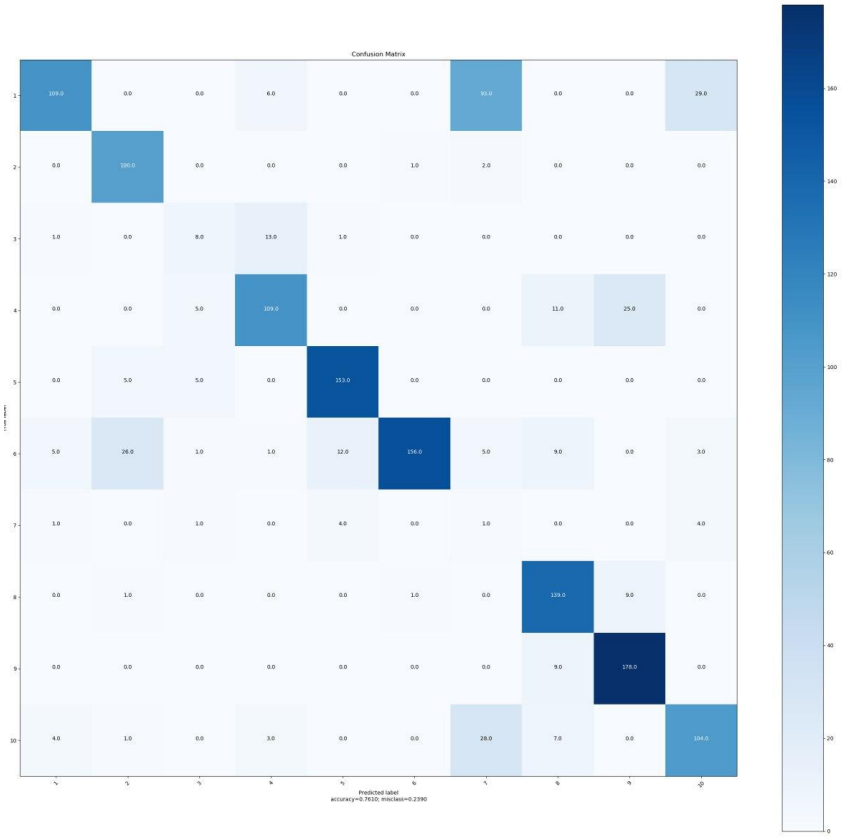


Figura B.14: Matriz de confusión por cada clase del dataset*Neuromorphic Benchmark* [15]. Configuración **n=5** y arquitectura de red ResNet50V2. Representación de eventos *por conjunto de frames*.

Tabla B.16: Precisión y ***Recall*** por cada clase del dataset propio. Configuración **n=5** y arquitectura de red ResNet50V2. Representación de eventos *por conjunto de frames*.

Clases	Precisión	***Recall***
Cruzar brazos	0.00	0.00
Levantarse	0.00	0.00
Coger del suelo	0.00	0.00
Andar	0.00	0.00
Saludar	0.30	0.30
Media	0.06	0.06

Tabla B.17: Precisión y ***Recall*** por cada clase del dataset propio. Configuración **n=5** y arquitectura de red ResNet50V2. Representación de eventos *por conjunto de frames*.

Clases	Precisión	***Recall***
Cruzar brazos	0.25	0.07
Levantarse	0.00	0.00
Coger del suelo	0.88	0.68
Andar	1.00	0.55
Saludar	1.00	0.20
Media	0.31	0.15

Tabla B.18: Precisión y ***Recall*** por cada clase del dataset propio. Configuración **n=5** y arquitectura de red ResNet50V2. Representación de eventos *por conjunto de frames*.

Clases	Precisión	***Recall***
Cruzar brazos	0.00	0.00
Levantarse	0.00	0.00
Coger del suelo	1.00	0.75
Andar	1.00	0.55
Saludar	0.00	0.00
Media	0.4	0.26

Apéndice C

Acciones del *Neuromorphic Benchmark*

En este apartado se muestran imágenes con los distintos movimientos que posee en el *data set* del título *Neuromorphic Benchmark* usado en los experimentos descritos en este trabajo:

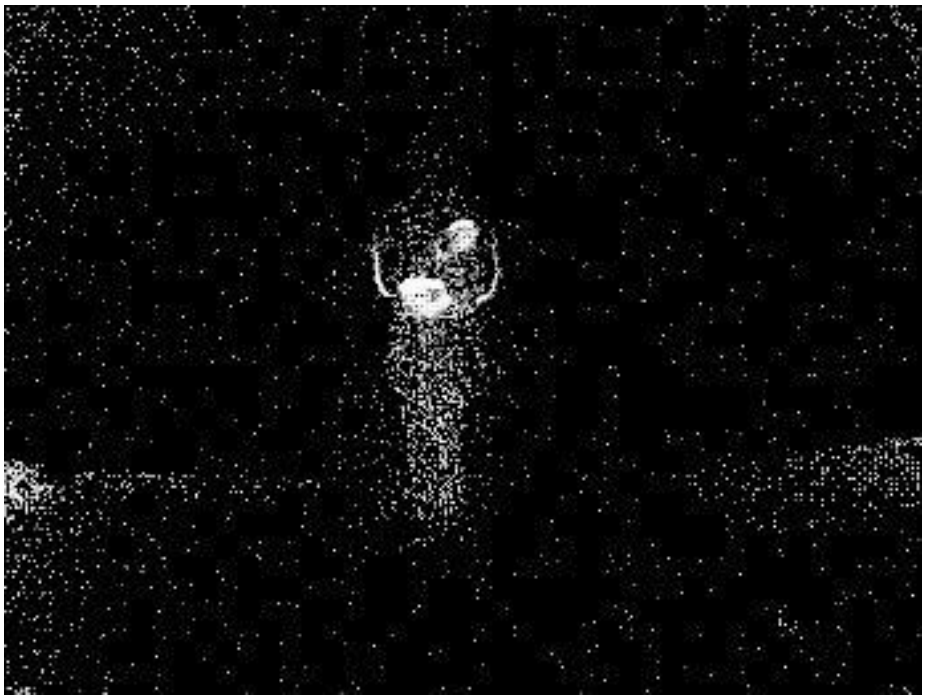


Figura C.1: *Frame* que muestra la acción cruzar los brazos del *data set Neuromorphic Benchmark* [15]

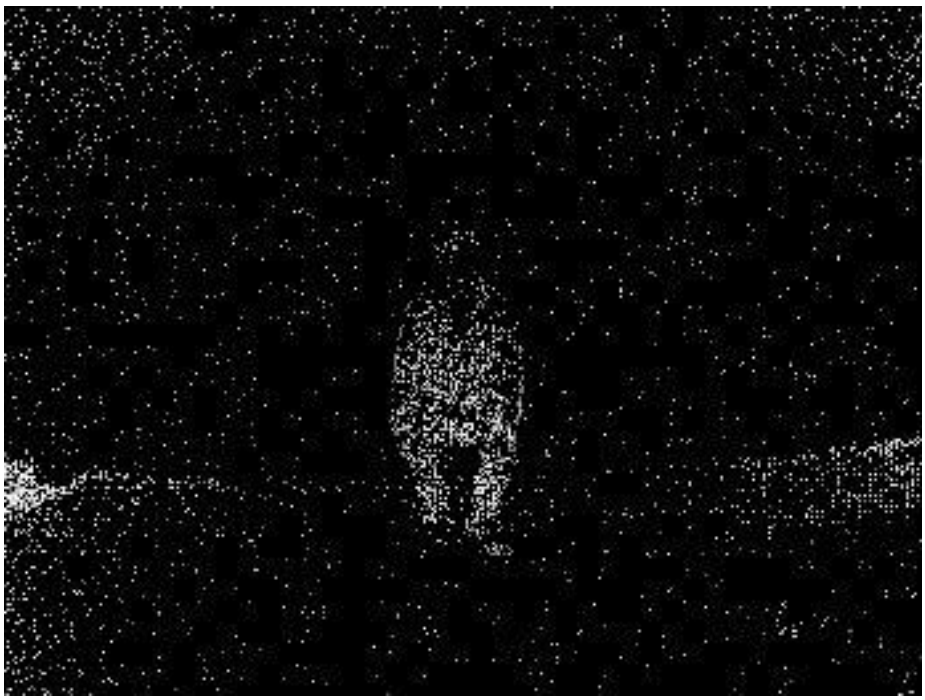


Figura C.2: *Frame* que muestra la acción levantarse del *data set Neuromorphic Benchmark* [15]

Figura C.3: *Frame* que muestra la acción saltar del *data set Neuromorphic Benchmark* [15]

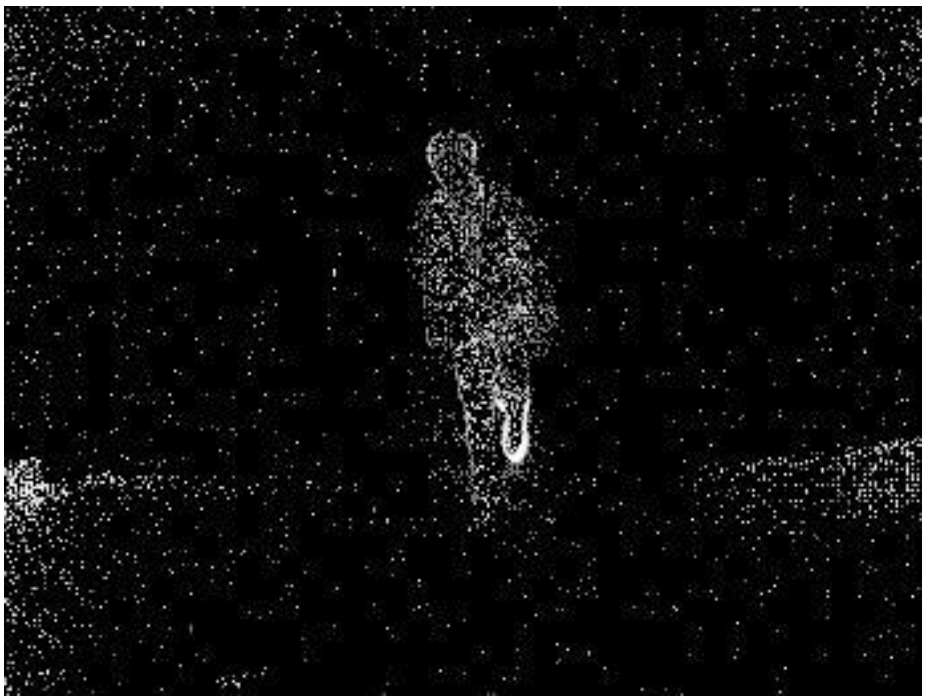


Figura C.4: *Frame* que muestra la acción lanzar una patada del *data set Neuromorphic Benchmark* [15]

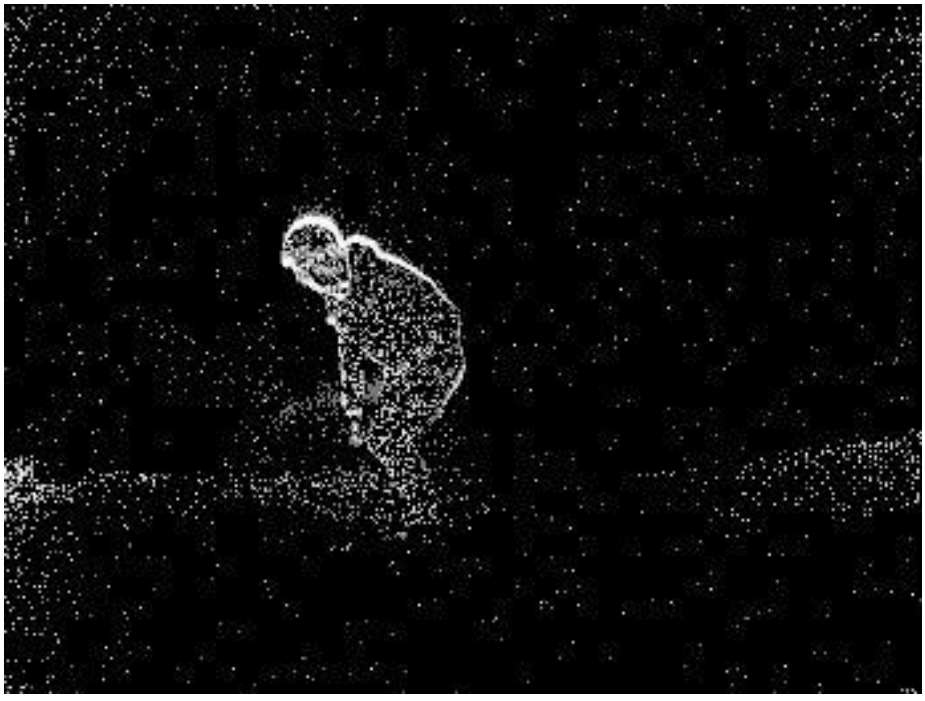


Figura C.5: *Frame* que muestra la acción coger algo del *data set Neuromorphic Benchmark* [15]

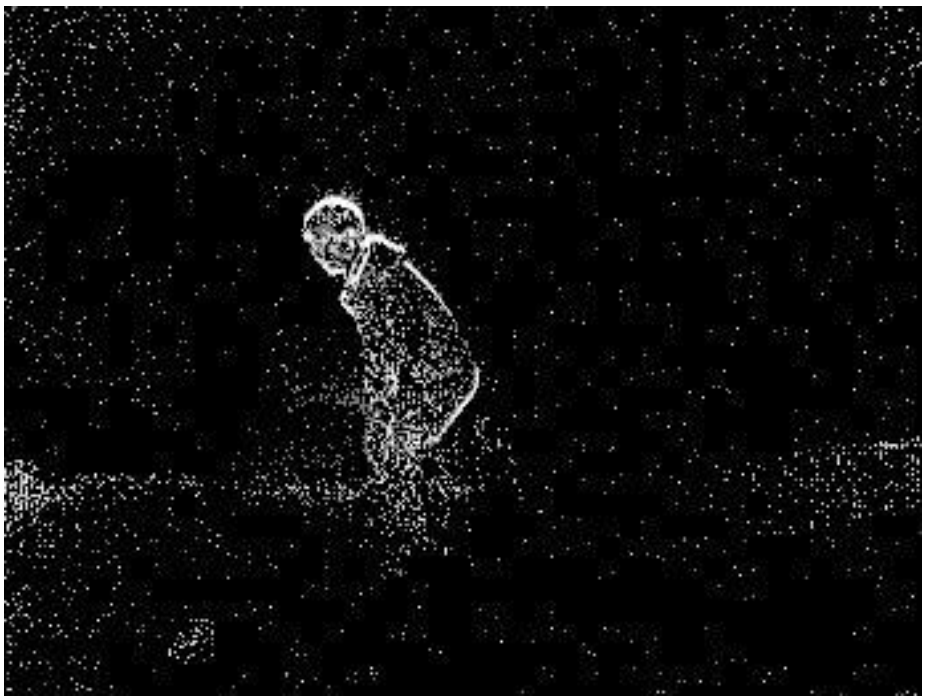


Figura C.6: *Frame* que muestra la acción sentarse del *data set Neuromorphic Benchmark* [15]

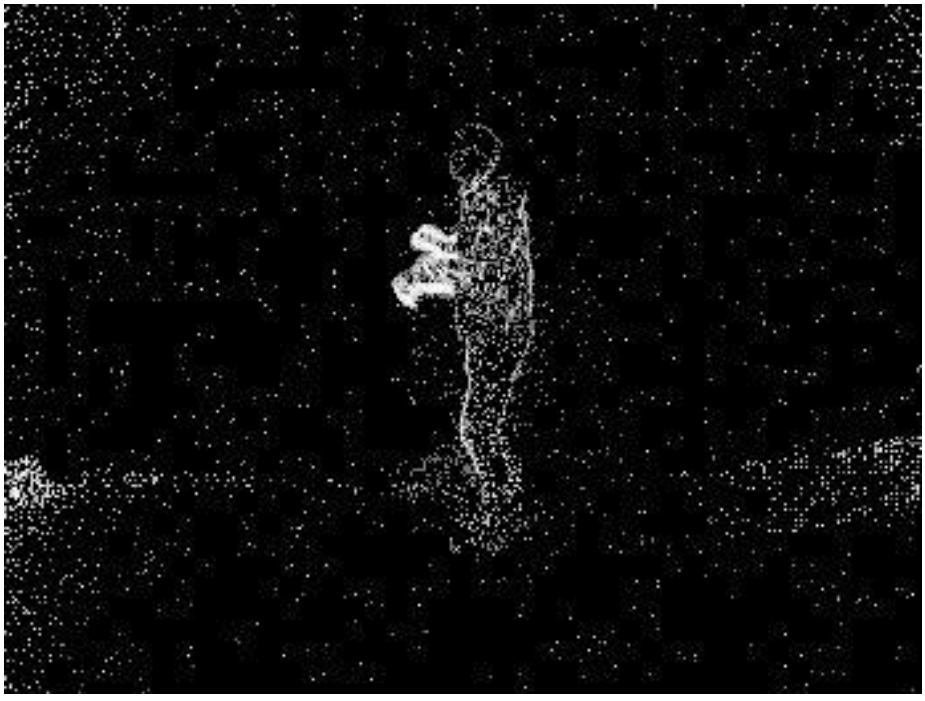


Figura C.7: *Frame* que muestra la acción lanzar del *data set Neuromorphic Benchmark* [15]

Figura C.8: *Frame* que muestra la acción darse la vuelta del *data set Neuromorphic Benchmark* [15]

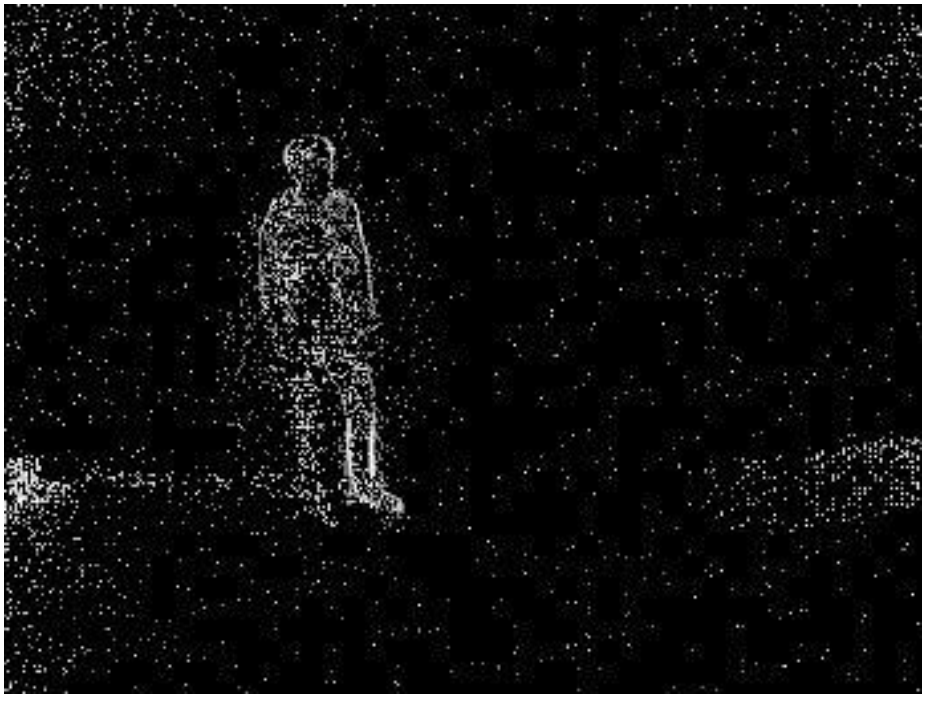


Figura C.9: *Frame* que muestra la acción caminar del *data set Neuromorphic Benchmark* [15]

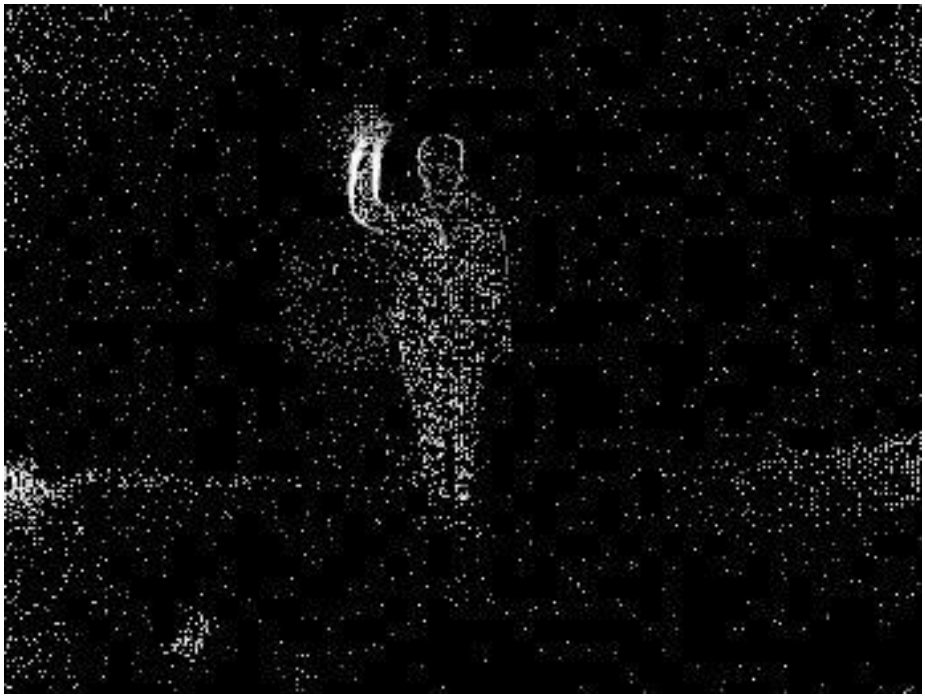


Figura C.10: *Frame* que muestra la acción saludar del *data set Neuromorphic Benchmark* [15]