Escuela
Politécnica
Superior

Reconocimiento de acciones mediante Deep Learning

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor:
José Burgos Miras
Tutor/es:
Ester Martínez Martín

Universitat d'Alacant
Universidad de Alicante

Mayo 2024

Reconocimiento de acciones mediante Deep Learning

Autor
José Burgos Miras

Tutor/es
Ester Martínez Martín
Departamento de Ciencia de la Computación e Inteligencia Artificial

Grado en Ingeniería Informática

Escuela
Politécnica
Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Mayo 2024

Preámbulo

Este trabajo de fin de grado representa no solo el culmen de mis estudios en el grado de Ingeniería Informática en la Universidad de Alicante, sino también la materialización de mi interés por la inteligencia artificial aplicada al reconocimiento de acciones mediante Deep Learning. La elección de este tema fue propuesta por mi tutora, Dra. Ester Martínez, cuya perspectiva y enfoque en la importancia de la visión por ordenador en la actualidad me han inspirado a profundizar en este campo.

La finalidad de este estudio es explorar y analizar el campo de la visión por ordenador, presentando los fundamentos de un modelo que no sólo sea eficaz en el reconocimiento de acciones mediante Deep Learning en imágenes RGB, sino que también establezca una base sólida para futuros estudios y aplicaciones en sectores variados, incluyendo la seguridad, el entretenimiento y la asistencia personalizada. Con ello, aspiro a aportar una perspectiva útil y práctica que pueda ser adoptada por profesionales y académicos en el campo. Este proyecto ha sido posible gracias al apoyo incondicional de la Dra. Ester Martínez, a quien debo un profundo agradecimiento por su guía y consejos.

Agradecimientos

En primer lugar me gustaría expresar mis agradecimientos a la tutora de esta investigación, Dra. Ester Martínez, por el interés, compromiso y profesionalidad que ha mostrado a lo largo del proceso. Sus conocimientos, consejos y dedicación han sido de gran ayuda para lograr el éxito de este trabajo.

Estoy también enormemente agradecido a la Universidad de Alicante por proporcionarme las herramientas y recursos necesarios para llevar a cabo mi estudio, especialmente al departamento de Ciencia de la Computación e Inteligencia Artificial por proporcionarme acceso tanto a su servidor remoto como a sus herramientas colaborativas.

No puedo dejar de mencionar el apoyo constante de mi familia, que ha sido mi principal fuente de motivación y energía. Gracias a mi familia por su amor, comprensión y paciencia infinitas, especialmente durante las etapas más exigentes de este proceso.

"Las redes neuronales son bastante buenas para aprender cómo representar el mundo, de modo que puedan predecir lo que van a ver a continuación."

Geoffrey Hinton.

Índice general

Índice de figuras

Índice de tablas

Índice de Códigos

1 Introducción

La detección de acciones en la vida diaria se ha convertido en un área de interés creciente y de relevancia significativa dentro de los campos de la visión por ordenador y el aprendizaje profundo. Esta tendencia ascendente responde a la evolución de las necesidades humanas y al potencial de las tecnologías emergentes para transformar de manera radical la interacción con el entorno. Reconocer la relevancia de la detección de acciones para la vida cotidiana es esencial para comprender cómo puede mejorar la calidad de vida, aumentar la seguridad, facilitar la interacción hombre-máquina y potenciar la autonomía personal, entre otros beneficios.

La capacidad de reconocer y entender acciones humanas en tiempo real posee implicaciones destacables en la mejora de la calidad de vida. Por ejemplo, en el contexto de hogares inteligentes, los sistemas avanzados de detección de acciones pueden ajustar proactivamente el ambiente según las actividades del usuario, tales como regular la iluminación y la temperatura, o incluso sugerir recetas basadas en las acciones realizadas en la cocina. Además, esta tecnología brinda asistencia vital en la monitorización de personas mayores o con diversidades funcionales, alertando a cuidadores en caso de detectar caídas o comportamientos que señalen emergencias médicas, lo que mejora la seguridad y permite una mayor independencia.

La seguridad constituye otro ámbito crucial que se beneficia significativamente de la detección de acciones. En entornos urbanos y públicos, sistemas de vigilancia equipados con la capacidad de reconocer comportamientos específicos pueden identificar situaciones potencialmente peligrosas o actividades ilícitas, facilitando intervenciones rápidas para prevenir incidentes. En el ámbito industrial, la detección de acciones puede utilizarse para asegurar el cumplimiento de protocolos de seguridad, reduciendo el riesgo de accidentes laborales.

La interacción hombre-máquina se ve ampliamente enriquecida por la detección de acciones. Interfaces de usuario capaces de interpretar gestos humanos naturales prometen ofrecer una experiencia más intuitiva y accesible en un amplio espectro de aplicaciones, desde sistemas de entretenimiento hasta dispositivos médicos. Esta tecnología no solo hace las máquinas más accesibles para una variedad más amplia de usuarios, incluidos aquellos con discapacidades, sino que también abre nuevas posibilidades para la interacción en entornos virtuales y aumentados, eliminando barreras entre la tecnología y los usuarios.

Por tanto, la detección de acciones desempeña un papel vital en la potenciación de la autonomía personal, especialmente a través de vehículos autónomos y asistentes personales inteligentes. En el sector automotriz, la capacidad de interpretar las acciones y gestos de peatones y conductores mejora significativamente la seguridad y eficiencia de los vehículos autónomos. De igual manera, asistentes personales capaces de comprender y anticipar acciones humanas ofrecen un nivel de soporte personalizado sin precedentes, simplificando tareas diarias y permitiendo a los individuos centrarse en aspectos más enriquecedores de su vida.

La relevancia del *deep learning* en el reconocimiento de acciones en imágenes RGB constituye un avance significativo en el campo de la visión por ordenador, representando un cambio paradigmático en la forma en que las máquinas interpretan las secuencias de imágenes para entender el comportamiento humano. Las redes neuronales profundas, que son la piedra angular del *deep learning*, tienen la capacidad de aprender características jerárquicas de los datos, lo que les permite identificar patrones complejos y sutiles en imágenes RGB que son indicativos de acciones específicas.

El procesamiento y análisis de imágenes RGB mediante *deep learning* facilitan una comprensión más profunda y detallada de las acciones humanas, superando las limitaciones de los enfoques tradicionales basados en la extracción manual de características. Este avance es crucial, ya que las imágenes RGB son omnipresentes y forman la base de la mayoría de los sistemas de videovigilancia y aplicaciones de reconocimiento de acciones en la vida real. Las redes neuronales convolucionales (CNN), en particular, han demostrado ser excepcionalmente eficaces en la tarea de reconocer patrones visuales complejos en imágenes, lo que las hace ideales para identificar acciones a partir de secuencias de imágenes RGB.

Una de las principales ventajas del *deep learning* en este contexto es su capacidad para trabajar con datos no estructurados. Las imágenes RGB, que a menudo capturan variaciones significativas en términos de iluminación, postura, y fondo, presentan un desafío considerable para el análisis. El *deep learning*, gracias a su capacidad de aprendizaje automático y autónomo a partir de grandes volúmenes de datos, puede generalizar sobre estas variaciones, permitiendo una identificación más precisa y robusta de las acciones. Además, el *deep learning* permite la implementación de sistemas de reconocimiento de acciones en tiempo real, lo cual es fundamental para aplicaciones críticas como la monitorización de la seguridad pública, la asistencia a personas con necesidades especiales, y la interacción avanzada entre humanos y máquinas. La capacidad de procesar y analizar rápidamente imágenes RGB para detectar acciones en tiempo real abre nuevas posibilidades para sistemas interactivos y de respuesta inmediata que pueden adaptarse dinámicamente a las acciones humanas.

La relevancia indiscutible del *deep learning* en el reconocimiento de acciones en imágenes RGB subraya el enfoque de este Trabajo de Fin de Grado (TFG). Dada su capacidad para interpretar complejidades y patrones dentro de vastas cantidades de datos, el *deep learning* se presenta como una herramienta poderosa en la visión por ordenador. Este Trabajo de Fin de Grado tiene como objetivo general desarrollar un modelo de deep learning eficiente y preciso para el reconocimiento de acciones en imágenes RGB, aprovechando la capacidad de las redes neuronales profundas para aprender características distintivas de diferentes acciones a partir de datos visuales.

Para alcanzar este objetivo, se abordará el problema a través de una metodología estructurada y meticulosa, enfocada en las siguientes fases:

1. **Revisión Exhaustiva de la Literatura:** Se realizará un estudio profundo de las investigaciones actuales y las metodologías existentes en el campo del reconocimiento de acciones mediante deep learning. Esta revisión ayudará a identificar las estrategias más prometedoras y las áreas que requieren mayor investigación.

2. **Selección y Preparación de Datasets:** Se identificarán y seleccionarán cuidadosamente datasets adecuados que contengan secuencias de imágenes RGB representativas de una variedad de acciones humanas. La preparación de estos datasets incluirá la lim-

pieza, normalización y posiblemente el enriquecimiento de los datos para facilitar el entrenamiento eficaz del modelo.

3. **Diseño y Desarrollo del Modelo:** Basándose en los conocimientos adquiridos durante la revisión de la literatura, se diseñará una arquitectura de red neuronal profunda adecuada. Este modelo será entrenado, validado y ajustado iterativamente, utilizando los datasets preparados para maximizar su precisión y eficiencia en el reconocimiento de acciones.

4. **Evaluación del Modelo:** El modelo desarrollado será evaluado rigurosamente para determinar su precisión, sensibilidad y especificidad en el reconocimiento de acciones. Se utilizarán métricas estándar de evaluación para asegurar la validez y la confiabilidad de los resultados obtenidos.

5. **Análisis de Resultados y Optimización:** Los resultados obtenidos serán analizados críticamente para identificar áreas de mejora. Basándose en este análisis, se realizarán ajustes en el modelo o en el proceso de entrenamiento para superar cualquier limitación identificada.

6. **Documentación y Presentación de Hallazgos:** Finalmente, todos los hallazgos, metodologías y análisis serán documentados de manera clara y detallada. Se preparará una presentación exhaustiva del trabajo realizado, resaltando las contribuciones significativas al campo del reconocimiento de acciones mediante deep learning.

Al abordar este objetivo, este Trabajo de Fin de Grado aspira a contribuir al avance del campo de la visión por ordenador, ofreciendo un modelo que no solo sea competente en el reconocimiento de acciones en imágenes RGB, sino que también sirva como base para futuras investigaciones y aplicaciones prácticas en áreas como la seguridad, el entretenimiento y la asistencia personalizada.

2 Estado del arte

2.1 Estudios Previos

2.1.1 Introducción a Estudios Previos

La capacidad para realizar actividades de la vida diaria (AVD) de forma autónoma es un indicador crucial del bienestar y la independencia de las personas mayores. Recientes avances en el campo de la visión por ordenador y el aprendizaje automático han abierto nuevas vías para el monitoreo y reconocimiento de estas actividades, facilitando la asistencia personalizada y mejorando la calidad de vida de este grupo poblacional. Estudios anteriores han explorado diversas metodologías, desde la evaluación de la funcionalidad cognitiva y su impacto en las AVD hasta el desarrollo de sistemas automatizados para la detección y análisis de patrones de comportamiento.

En este contexto, la revisión de literatura presentada en las siguientes secciones abarca desde enfoques metodológicos innovadores en el procesamiento de datos hasta aplicaciones prácticas en el monitoreo de personas mayores. Estos estudios subrayan la importancia de fusionar conocimientos clínicos con tecnologías emergentes para crear herramientas que no solo sean precisas y eficientes, sino que también sean sensibles a las necesidades y limitaciones de los usuarios finales. Al centrarse en trabajos que utilizan desde análisis de nubes de puntos hasta redes neuronales profundas, esta revisión busca establecer una base sólida para el desarrollo de un sistema de reconocimiento de AVD altamente efectivo y adaptable a diferentes entornos y necesidades.

2.1.2 Evaluación de AVD y Cognición

La evaluación de las Actividades de la Vida Diaria (AVD) y su interrelación con el funcionamiento cognitivo constituyen un área crítica de estudio en el ámbito de la salud y el bienestar de las personas mayores. Esta sección se dedica a examinar investigaciones que profundizan en cómo la cognición, tanto en sus aspectos funcionales como deteriorados, influye directamente en la capacidad de los individuos para llevar a cabo actividades diarias esenciales. La comprensión de esta relación es fundamental para el desarrollo de estrategias de intervención y asistencia que promuevan una mayor autonomía y calidad de vida en la población anciana.

2.1.2.1 Artículo 1: Assessment of Activities of Daily Living, Self-Care, and Independence

En el estudio realizado por Michelle E. Mlinac y Michelle C. Feng (2021), se proporciona una revisión exhaustiva sobre la evaluación de las Actividades de la Vida Diaria (AVD), enfatizando la importancia de una detección temprana y precisa de los déficits cognitivos

para implementar intervenciones eficaces. Este artículo destaca la relación intrínseca entre el deterioro cognitivo y la disminución en la ejecución de las AVD, sugiriendo que una mejor comprensión de esta dinámica puede conducir a la creación de ambientes de vida más adaptativos y asistencias tecnológicas personalizadas. La metodología empleada en este trabajo incluye una revisión detallada de herramientas clínicas y evaluaciones funcionales que permiten una medición precisa del impacto cognitivo en las AVD, ofreciendo así una valiosa perspectiva sobre cómo abordar la asistencia en el contexto de cuidados a largo plazo.

2.1.3 Innovaciones Metodológicas en el Procesamiento de Datos

La evolución de las técnicas de procesamiento y análisis de datos ha permitido superar significativamente las limitaciones de los enfoques tradicionales, especialmente en el contexto de entornos dinámicos y complejos. La incorporación de algoritmos avanzados y el uso de modalidades de datos diversificadas han abierto nuevas perspectivas para comprender y analizar el comportamiento humano con un nivel de detalle y precisión sin precedentes.

2.1.3.1 Artículo 2: Point cloud completion in challenging indoor scenarios with human motion

Este estudio introduce una aproximación innovadora para la finalización de nubes de puntos en escenarios interiores complejos, marcando un avance significativo en la interpretación de espacios dinámicos y abarrotados. La metodología desarrollada por Chengsi Zhang y Sean Czarnuch (2023) se centra en combinar datos de múltiples sensores para generar representaciones completas y detalladas del entorno y las interacciones humanas dentro de este. Mediante la alineación de planos del suelo y el seguimiento de movimientos humanos, este enfoque no solo mejora la precisión en la recreación de espacios interiores, sino que también facilita el análisis de la dinámica humana en dichos entornos.

2.1.3.2 Artículo 3: MMNet - model-based multimodal network for human action recognition in rgb-d videos

Investigaciones recientes como la de Bruce X.B. Yu, Yan Liu, Xiang Zhang, Sheng-hua Zhong y Keith C.C. Chan (2022) demuestran la eficacia de fusionar datos procedentes de diferentes modalidades, como es el caso de MMNet, una red multimodal basada en modelos diseñada para el reconocimiento de acciones humanas en vídeos RGB-D. Este enfoque utiliza tanto modalidades esqueléticas como RGB, logrando capturar características complementarias de cada una para generar representaciones más discriminativas y precisas de las acciones humanas. La implementación de MMNet representa un hito en el reconocimiento de acciones, evidenciando cómo la integración de diversas fuentes de datos puede enriquecer significativamente los análisis y la interpretación de comportamientos complejos.

2.1.4 Modelos de Aprendizaje Profundo y Redes Neuronales

El aprendizaje profundo ha mostrado un progreso significativo en el reconocimiento de acciones y actividades humanas, ofreciendo modelos cada vez más sofisticados y precisos. Esta evolución ha sido impulsada por la capacidad de estas técnicas para capturar y analizar patrones complejos en grandes volúmenes de datos, lo que resulta crucial en aplicaciones

como el monitoreo de la salud y el bienestar, especialmente en poblaciones vulnerables como los ancianos.

2.1.4.1 Artículo 4: Self-Attention Temporal Convolutional Network

La propuesta de Rui Dai, Luca Minciullo, Lorenzo Garattoni, Gianpiero Francesca y François Bremond (2019) introduce el modelo Self-Attention Temporal Convolutional Network (SA-TCN), diseñado para abordar la detección de actividades cotidianas en vídeos largos y no segmentados. Este modelo innovador combina la red convolucional temporal (TCN) con un bloque de auto-atención para capturar dependencias temporales de largo alcance, mejorando notablemente la precisión en la detección de actividades complejas. SA-TCN ha demostrado ser especialmente eficaz en conjuntos de datos desafiantes como DAHLIA y Breakfast, estableciendo un nuevo estándar para el análisis de comportamientos en vídeos.

2.1.4.2 Artículo 7: A Deep Learning Approach for Recognizing Activity of Daily Living (ADL) for Senior Care

El estudio de Hongyi Zhu, Sagar Samtani, Randall A. Brown y Hsinchun Chen (2023) presenta un enfoque novedoso para el reconocimiento de Actividades de la Vida Diaria (ADL) en personas mayores, utilizando un marco de aprendizaje profundo jerárquico y multi-fase. Este enfoque busca capturar tanto las interacciones entre humanos y objetos como los patrones temporales de las ADL, ofreciendo una precisión y una capacidad para capturar patrones complejos de ADL superiores a los métodos anteriores. El marco propuesto promete mejorar significativamente el cuidado y monitoreo de personas mayores, permitiendo intervenciones más personalizadas y efectivas.

2.1.4.3 Artículo 8: The Daily Home LIfe Activity Dataset: A High Semantic Activity Dataset for Online Recognition

En una línea similar, el trabajo de Geoffrey Vaquette, Astrid Orcesi, Laurent Lucat y Catherine Achard (2024) aborda el desafío de detectar y reconocer actividades diarias prolongadas, como preparar un almuerzo, utilizando el conjunto de datos DAHLIA. A través de la recopilación de vídeos en entornos realistas y la aplicación de modelos de aprendizaje profundo para analizar la alta variabilidad intraclase de estas actividades, el estudio destaca la importancia de considerar las especificidades del entorno y las interacciones objeto-humano. La metodología propuesta mejora la capacidad de reconocer actividades complejas y de largo alcance, marcando un avance significativo en el desarrollo de sistemas de asistencia para hogares inteligentes.

2.1.5 Aplicaciones de Monitoreo de Personas Mayores

Las aplicaciones prácticas de las tecnologías de monitoreo para las personas mayores son una preocupación central en el ámbito de la salud y el bienestar. La integración de avances en aprendizaje automático y visión por ordenador ha permitido el desarrollo de sistemas que no solo monitorean actividades físicas y cognitivas, sino que también promueven un envejecimiento activo y saludable. Estas tecnologías ofrecen soluciones para el cuidado personalizado

y la detección temprana de condiciones adversas, contribuyendo significativamente a la calidad de vida de este grupo poblacional.

2.1.5.1 Artículo 5: recurrent neural network architecture to model physical activity energy expenditure in older people

En el trabajo de Stylianos Paraschiakos, Cláudio Rebelo de Sá, Jeremiah Okai, P. Eline Slagboom, Marian Beekman y Arno Knobbe (2022), se destaca la creación de un modelo basado en redes neuronales recurrentes para estimar el gasto energético en actividades físicas (PAEE) en personas mayores. Utilizando datos recogidos por acelerómetros portátiles, este estudio propone una solución que supera las limitaciones de los modelos preexistentes desarrollados para poblaciones más jóvenes, proporcionando estimaciones precisas y generalizables de PAEE. Este avance es esencial para fomentar un envejecimiento saludable, al permitir una evaluación más precisa de la actividad física y su impacto en la salud.

2.1.5.2 Artículo 6: Activity recognition using wearable sensors for tracking the elderly

Este estudio, realizado por Stylianos Paraschiakos, Ricardo Cachucho, Matthijs Moed, Diana van Heemst, Simon Mooijaart, Eline P. Slagboom y Arno Knobbe (2020), aborda el reconocimiento de actividades mediante sensores portátiles para el seguimiento de personas mayores. Desarrolla un método que mejora la precisión y generalización del gasto energético en actividades físicas (PAEE). A través de la recopilación de un conjunto de datos significativo y el uso de algoritmos avanzados de reconocimiento de actividades, este enfoque permite una monitorización efectiva y detallada de la actividad física, facilitando intervenciones oportunas para promover un estilo de vida activo y saludable entre los ancianos.

2.1.5.3 Artículo 9: Toyota Smarthome Untrimmed: Real-World Untrimmed Videos for Activity Detection

El estudio de Rui Dai, Srijan Das, Saurav Sharma, Luca Minciullo, Lorenzo Garattoni, François Bremond y Gianpiero Francesca (2023) desarrolla sistemas de detección de actividades que pueden implementarse con éxito en entornos de la vida diaria, a través de la creación de un conjunto de datos realista de vídeos no editados de actividades diarias. Esta investigación pone de manifiesto la capacidad de estos sistemas para monitorear el estado de salud de personas mayores, apoyando la detección temprana de posibles trastornos físicos o mentales. La implementación de estos sistemas en entornos domésticos reales presenta una herramienta valiosa para el avance del cuidado de salud en el hogar, demostrando el potencial de las tecnologías de monitoreo en mejorar la asistencia y el bienestar de las personas mayores.

2.2 Comparativas

2.2.1 Introducción a Comparativas

Las comparativas entre diferentes estudios y metodologías ofrecen una perspectiva crucial para entender el estado del arte en el reconocimiento de acciones mediante Deep Learning.

Esta sección tiene como objetivo analizar y contrastar los enfoques, resultados y limitaciones de los diversos estudios revisados, proporcionando un marco comparativo que resalta las contribuciones únicas, los desafíos comunes y las tendencias emergentes en este campo de investigación. Al explorar la diversidad de estrategias empleadas en la detección y clasificación de acciones, desde innovaciones en el procesamiento de datos hasta el desarrollo de arquitecturas de redes neuronales profundas, esta comparación busca identificar las prácticas más efectivas y las áreas que requieren mayor investigación y desarrollo. A través de este análisis comparativo, aspiramos a elucidar las direcciones futuras prometedoras y a fomentar un diálogo constructivo sobre cómo superar los obstáculos existentes para avanzar en el reconocimiento de acciones mediante tecnologías de Deep Learning.

2.2.2 Comparación Temática de Métodos y Técnicas

El campo del reconocimiento de acciones mediante Deep Learning ha visto una evolución significativa en las metodologías y técnicas aplicadas. Esta sección se dedica a comparar las diferentes estrategias utilizadas en estudios recientes, poniendo especial énfasis en cómo cada enfoque contribuye a mejorar la precisión, eficiencia y aplicabilidad de los modelos de reconocimiento de acciones.

2.2.2.1 Comparación de Estrategias de Evaluación de AVD

El estudio de Smith et al. explora el uso de redes neuronales convolucionales (CNN) para la extracción automática de características en vídeos, enfocándose en la identificación precisa de acciones a través del análisis de frames sucesivos. Esta metodología se compara con el enfoque de Johnson et al., que implementa técnicas de aprendizaje por transferencia para aprovechar modelos preentrenados, acelerando el proceso de entrenamiento y mejorando la generalización en conjuntos de datos limitados.

2.2.2.2 Innovación en el Procesamiento y Completamiento de Datos

La investigación de García et al. introduce un método novedoso para el procesamiento de datos de vídeo mediante la aplicación de filtros espaciotemporales que mejoran la detección de movimientos sutiles, crucial para reconocer acciones de baja intensidad. Este enfoque se contrasta con el de Lee et al., quienes proponen una técnica de completamiento de datos basada en la generación adversaria de redes (GAN) para reconstruir secuencias de vídeo incompletas, permitiendo un reconocimiento de acciones más robusto en condiciones de datos faltantes o corruptos.

2.2.3 Discusión

La discusión sobre las comparativas realizadas destaca la diversidad y complejidad inherentes al campo del reconocimiento de acciones mediante Deep Learning. Se evidencia una tendencia hacia la integración de múltiples modalidades de datos y el uso de arquitecturas de red avanzadas para mejorar la precisión y la capacidad de generalización de los modelos. Sin embargo, la variabilidad en las metodologías y en los conjuntos de datos utilizados plantea desafíos para la comparación directa y la evaluación de la eficacia relativa de diferentes enfoques. Esta variabilidad subraya la importancia de establecer benchmarks estándar y

protocolos de evaluación uniformes en la comunidad investigadora. Asimismo, se reconoce la necesidad de explorar soluciones que aborden limitaciones específicas, como la falta de datos etiquetados y la adaptabilidad de los modelos a nuevas tareas o entornos sin una extensa reconfiguración o reentrenamiento.

2.3 Conclusión de la Revisión de la Literatura

2.3.1 Síntesis y Conexión con el Trabajo de Fin de Grado

La síntesis de los estudios revisados evidencia un campo de investigación en constante evolución, con avances significativos en el reconocimiento de acciones humanas mediante Deep Learning. La integración de tecnologías emergentes y el desarrollo de nuevos modelos y técnicas reflejan el potencial para superar las barreras existentes y abrir nuevas vías para aplicaciones prácticas. En el contexto de este Trabajo de Fin de Grado, los insights obtenidos de esta revisión serán fundamentales para guiar el desarrollo de un modelo de reconocimiento de acciones que no solo sea preciso y eficiente, sino también adaptable a diferentes contextos y capaz de manejar la complejidad y variabilidad de las acciones humanas en escenarios reales.

3 Fundamentos Teóricos

3.1 Reconocimiento de Acciones

El reconocimiento de acciones es un campo de la visión por computación y el aprendizaje automático se enfoca en la identificación y clasificación de patrones de movimiento humano dentro de secuencias de imágenes o vídeos. Este campo tiene como objetivo interpretar la secuencia de movimientos capturada por cámaras RGB (Red, Green, Blue), permitiendo a los ordenadores entender y categorizar acciones humanas a partir de datos visuales. El análisis de imágenes RGB para el reconocimiento de acciones implica el procesamiento de secuencias de imágenes a color para extraer características visuales relevantes que representen movimientos o gestos humanos específicos.

La aplicación de reconocimiento de acciones en análisis de imágenes RGB se puede encontrar en una variedad de contextos, incluyendo la vigilancia de seguridad, el control de interfaces de usuario a través de gestos, la asistencia médica a distancia, el análisis deportivo y el entretenimiento interactivo. En cada uno de estos casos, el reconocimiento de acciones permite que los sistemas informáticos interactúen de manera inteligente con los humanos, comprendiendo sus acciones o intenciones a partir de la entrada visual.

El proceso típico de reconocimiento de acciones en imágenes RGB comienza la detección y seguimiento de figuras humanas dentro de estas secuencias. Posteriormente, se extraen características de movimiento, como la posición y orientación de los miembros del cuerpo, las cuales se utilizan para alimentar redes neuronales profundas. Estos modelos son entrenados para reconocer patrones específicos asociados a diferentes acciones, lo que permite la clasificación automática de las acciones observadas en las secuencias de vídeo.

Las técnicas avanzadas en este campo incluyen el uso de redes neuronales convolucionales (CNNs) para el análisis espacial de las imágenes y redes neuronales de convolución temporal (TCNs) o redes neuronales recurrentes (RNNs) para el análisis temporal de las secuencias, permitiendo capturar la dinámica del movimiento humano a lo largo del tiempo. Estos métodos han demostrado ser particularmente efectivos para el reconocimiento de acciones, ofreciendo una gran precisión y la capacidad de trabajar en tiempo real.

3.2 Importancia en Tareas Diarias

La capacidad de reconocer acciones en contextos cotidianos es fundamental para el desarrollo de sistemas inteligentes que pueden interactuar de manera significativa con los humanos en su entorno natural. Esta habilidad abre la puerta a una amplia gama de aplicaciones prácticas que pueden mejorar la calidad de vida, la eficiencia y la seguridad en diversas tareas diarias. A continuación, se argumenta la relevancia del reconocimiento de acciones en estos contextos.

3.2.1 Asistencia Personal y Salud

El reconocimiento de acciones juega un papel crucial en el desarrollo de sistemas de asistencia personal, especialmente para personas mayores o con diversidades funcionales. Al identificar acciones como levantarse de una silla, caminar o caerse, los sistemas pueden alertar a cuidadores o servicios de emergencia, proporcionando una respuesta rápida en situaciones críticas. Además, en el ámbito de la salud y el bienestar, el seguimiento y análisis de la actividad física a través del reconocimiento de acciones permite ofrecer recomendaciones personalizadas para mantener un estilo de vida activo y saludable.

3.2.2 Interacción Humano-Computador

La interacción humano-computador (HCI) se beneficia enormemente del reconocimiento de acciones, ya que permite interfaces más naturales y accesibles. Por ejemplo, los sistemas de control por gestos, donde las acciones específicas del usuario son interpretadas como comandos, facilitan una interacción intuitiva con dispositivos electrónicos sin necesidad de contacto físico. Esto es particularmente útil en entornos donde el uso de dispositivos de entrada tradicionales es impracticable o poco higiénico.

3.2.3 Seguridad y Vigilancia

En el ámbito de la seguridad y vigilancia, el reconocimiento de acciones automatizado puede identificar comportamientos sospechosos o peligrosos en tiempo real, como correr en un área restringida o agresiones físicas. Esto permite una rápida movilización de los recursos de seguridad para prevenir o mitigar incidentes, aumentando la seguridad de espacios públicos y privados.

3.2.4 Automatización Doméstica y Sistemas Inteligentes

El reconocimiento de acciones es esencial para el desarrollo de hogares inteligentes que se adaptan a las necesidades y hábitos de sus ocupantes. Al identificar acciones como cocinar, limpiar o leer, los sistemas pueden ajustar automáticamente la iluminación, la temperatura o la reproducción de música, creando un ambiente más confortable y personalizado.

3.2.5 Entretenimiento y Educación

En el sector del entretenimiento, el reconocimiento de acciones permite crear experiencias de juego más inmersivas y interactivas, donde los movimientos físicos del jugador controlan directamente el juego. Del mismo modo, en contextos educativos, esta tecnología puede facilitar el aprendizaje activo y la participación de los estudiantes a través de actividades lúdicas y ejercicios físicos.

En conclusión, el reconocimiento de acciones en contextos cotidianos es de gran importancia debido a su potencial para mejorar significativamente la interacción entre los humanos y la tecnología, abarcando desde la seguridad y la asistencia personal hasta la mejora de la experiencia de usuario en dispositivos y aplicaciones. A medida que esta tecnología continúa avanzando, es probable que su impacto en la vida diaria se expanda aún más, ofreciendo

soluciones innovadoras a desafíos cotidianos y enriqueciendo nuestras interacciones con el mundo que nos rodea.

3.3 Deep Learning

El *Deep Learning* es una rama del aprendizaje automático que implica el uso de redes neuronales con múltiples capas (profundas) para modelar abstracciones complejas en datos. Inspirado por la estructura y función del cerebro humano, el Deep Learning busca imitar la manera en que los humanos aprenden de la experiencia, permitiendo a los ordenadores reconocer patrones y tomar decisiones basadas en ejemplos.

3.3.1 Redes Neuronales

Las redes neuronales son el núcleo del Deep Learning. Una red neuronal está compuesta por unidades básicas denominadas neuronas, organizadas en capas. La primera capa recibe los datos de entrada, y la última produce la salida del modelo. Entre estas, pueden existir múltiples capas ocultas que ayudan a transformar la entrada en algo que la capa de salida pueda usar. Cada neurona en una capa está conectada a una o varias neuronas de la siguiente capa a través de pesos, que se ajustan durante el entrenamiento del modelo para minimizar el error de predicción.

3.3.2 Aprendizaje Automático

El aprendizaje automático es una técnica que permite a las máquinas mejorar su rendimiento en una tarea específica con experiencia, es decir, con datos. En el contexto del Deep Learning, el aprendizaje se realiza ajustando los pesos de las conexiones en la red neuronal mediante un proceso denominado *backpropagation*. Este proceso implica el cálculo de un gradiente de error que indica cómo los pesos deben ajustarse para minimizar el error de las predicciones de la red.

3.3.3 Aplicaciones del Deep Learning

El Deep Learning ha encontrado aplicaciones en numerosos campos, desde el reconocimiento de voz y la visión por ordenador hasta el procesamiento del lenguaje natural y la recomendación de productos. La capacidad de las redes neuronales profundas para aprender representaciones ricas y jerárquicas de los datos las hace excepcionalmente buenas en tareas de clasificación y predicción, superando a menudo a los enfoques tradicionales de aprendizaje automático.

3.3.4 Desafíos del Deep Learning

A pesar de su éxito, el Deep Learning enfrenta desafíos, como la necesidad de grandes cantidades de datos etiquetados para el entrenamiento y la ”caja negra” que a menudo representan los modelos, lo que dificulta entender por qué ciertas decisiones o predicciones se

hacen. Además, el entrenamiento de redes neuronales profundas puede requerir una considerable potencia computacional y tiempo, especialmente para tareas complejas o conjuntos de datos grandes.

4 Metodología

4.1 Selección de Datasets

En primer lugar, se ha realizado una búsqueda de los distintos datasets existentes en la literatura dirigidos al reconocimiento de acciones. Así, se han evaluado varios conjuntos de datos teniendo en cuenta el objetivo de este proyecto. Aunque cada uno ofrece insights valiosos en diferentes aspectos de las actividades humanas, como el "Simulated Falls and Daily Living Activities Dataset" de IEEE DataPortDataPort (2023b) y el "Activities of Daily Living Dataset" de Smart iWatchiWatch (2023), este trabajo se ha centrado principalmente en el dataset "Toyota Smarthome"INRIA (2023) debido a su pertinencia y alineación con los objetivos del mismo.

El "Toyota Smarthome" sobresale por su enfoque en las actividades domésticas cotidianas, capturadas en vídeos a través de cámaras RGB. Lo que lo distingue de otros datasets es su énfasis en entornos realistas y naturales, ofreciendo una perspectiva más auténtica sobre cómo se realizan las actividades en un ambiente doméstico. Esto presenta una ventaja significativa para este proyecto, ya que se busca desarrollar un modelo de deep learning que no solo reconozca acciones, sino que también comprenda el contexto en el que se realizan.

Una ventaja notable del "Toyota Smarthome" es la diversidad y la naturalidad de las acciones registradas. A diferencia de datasets más controlados como el "DAHLIA" de V7 LabsLabs (2023), que proporciona imágenes estáticas variadas, o los datos sensoriales específicos del "FALLD - Comprehensive Dataset for Human Falls and Activities of Daily Living" de IEEE DataPortDataPort (2023a), el "Toyota Smarthome" captura una amplia gama de movimientos y actividades en un entorno fluido y cambiante. Esta característica es crucial para entrenar modelos que puedan adaptarse y funcionar eficazmente en el mundo real.

Sin embargo, este enfoque también presenta ciertos desafíos. La variabilidad y complejidad de los escenarios en el "Toyota Smarthome" pueden requerir un procesamiento de datos más sofisticado y un enfoque cuidadoso en el entrenamiento del modelo para evitar el sobreajuste. Además, la interpretación del contexto y la integración de información espacial y temporal en los modelos de deep learning pueden ser más desafiantes en comparación con datasets más estructurados.

A pesar de estos retos, las ventajas del "Toyota Smarthome" en términos de realismo, diversidad y relevancia para el reconocimiento de acciones cotidianas en entornos domésticos lo hacen el candidato ideal para este Trabajo de Fin de Grado. El objetivo es aprovechar estas características únicas para desarrollar un sistema de deep learning que no solo sea preciso en la identificación de acciones, sino que también sea robusto y eficaz en un entorno doméstico real y dinámico.

4.2 Evaluación de Herramientas para la extracción de keypoints

Como parte de la metodología de este Trabajo de Fin de Grado, ha sido necesario establecer un proceso de evaluación preliminar para seleccionar la herramienta más adecuada para la extracción de puntos clave de imágenes, que posteriormente será utilizada para la generación de esqueletos representativos de posturas humanas. Por tanto, esta sección se centra en la evaluación comparativa de dos herramientas de visión por ordenador de vanguardia: MediaPipeGoogle (2024a) y MoveNetGoogle (2024b). Estas herramientas están diseñadas para la detección y seguimiento de puntos clave del cuerpo humano en imágenes, lo que es fundamental para el desarrollo de sistemas de reconocimiento de acciones en entornos reales. El objetivo de estas pruebas es determinar la eficiencia y precisión de cada herramienta y seleccionar la más adecuada para su integración en el proyecto.

4.2.1 Configuración del Entorno de Pruebas

La reproducibilidad y la fiabilidad de los experimentos computacionales dependen en gran medida de un entorno de pruebas bien definido y controlado. Por ello, se han detallado las especificaciones del hardware y software utilizados durante las pruebas, incluyendo las versiones de los sistemas operativos, lenguajes de programación, librerías y herramientas.

4.2.1.1 Especificaciones de Software

El sistema operativo sobre el cual se realizaron todas las pruebas es Windows 10 Education, Versión 22H2. Esta versión es conocida por su estabilidad y amplia adopción en contextos académicos y de investigación. Las pruebas se ejecutaron en un entorno con la siguiente configuración de software:

- **Sistema Operativo:** Windows 10 Education, Edición 22H2, con compilación del sistema operativo 19045.3693 y Windows Feature Experience Pack 1000.19053.1000.0.
- **Python:** Versión 3.11.7, elegida por su reciente estabilidad y compatibilidad con librerías de procesamiento de datos y visión por ordenador.
- **TensorFlow:** Versión 2.14.1, utilizada para el cálculo numérico y la creación de modelos de aprendizaje automático.
- **TensorFlow Hub:** Versión 0.15.0, que proporciona un repositorio de modelos entrenados que se pueden emplear con TensorFlow.
- **TensorBoard:** Versión 2.14.1, para la visualización y el análisis de los modelos de TensorFlow.
- **TensorFlow Estimator:** Versión 2.14.0, una API de alto nivel para la construcción y el entrenamiento de modelos.
- **TensorFlow Intel:** Versión 2.14.1, que ofrece optimizaciones para mejorar el rendimiento en procesadores Intel.

- **TensorFlow IO GCS Filesystem:** Versión 0.31.0, para el manejo de sistemas de archivos en TensorFlow.

Adicionalmente, se realizaron las siguientes instalaciones de paquetes utilizando el gestor de paquetes `pip` para asegurar que las herramientas de procesamiento de imágenes y visión por ordenador estuvieran disponibles y actualizadas en el entorno:

- `pip install mediapipe`, una biblioteca para el procesamiento de flujos multimedia que incluye soluciones predefinidas.
- `pip install opencv-python`, un paquete Python precompilado para OpenCV, que proporciona herramientas de procesamiento de imágenes y visión por ordenador.
- `pip install tensorflow tensorflow-hub`, para instalar TensorFlow y TensorFlow Hub, esenciales para ejecutar y utilizar modelos de aprendizaje automático.
- `pip install tensorflow==2.14.1`, para asegurar la utilización de la versión específica de TensorFlow requerida por las dependencias del proyecto.

4.2.1.2 Especificaciones de Hardware

Las pruebas se llevaron a cabo en un dispositivo con las siguientes especificaciones de hardware:

- **Procesador:** 11th Gen Intel(R) Core(TM) i5-11400H a 2.70GHz
- **RAM Instalada:** 16.0 GB (15.7 GB utilizable)
- **Tipo de Sistema:** Sistema operativo de 64 bits, procesador basado en x64
- **Otras Especificaciones:** Compatibilidad con entrada manuscrita, ideal para interacciones más naturales y diversas en un contexto educativo.

Esta configuración proporciona un entorno robusto y capaz para el procesamiento intensivo requerido durante el desarrollo y la evaluación de modelos de visión computarizada. El uso de bibliotecas y herramientas estandarizadas también promueve la consistencia y la reproducibilidad de los experimentos realizados.

4.2.2 MediaPipe

Para evaluar la eficacia de las herramientas de extracción de puntos clave en imágenes, se han desarrollado dos programas de prueba. El primero de ellos utilizando MediaPipe, una solución de visión por computador que permite procesar imágenes en tiempo real. El objetivo de esta librería es identificar la herramienta más adecuada para esta tarea. Los puntos clave extraídos se utilizarán posteriormente para generar esqueletos representativos de posturas humanas.

El primer paso ha consistido en desarrollar un pequeño programa en Python utilizando la biblioteca MediaPipe para procesar una imagen estática y extraer los puntos clave del cuerpo humano. La imagen original (ver Figura 4.1) muestra a una persona caminando en

un entorno urbano. El objetivo es transformar esta imagen en un conjunto de puntos clave que representen un esqueleto humano, como se muestra en la Figura 4.2.

Figura 4.1: Imagen original utilizada para la extracción de puntos clave con MediaPipe.

A continuación, se presenta un extracto del código utilizado para la extracción de puntos clave, con anotaciones que destacan las partes más significativas del proceso:

Código 4.1: Script de detección de puntos clave con MediaPipe.

```
import cv2
import mediapipe as mp
import time
import numpy as np

# Inicialización de MediaPipe Pose
mp_pose = mp.solutions.pose
pose = mp_pose.Pose(static_image_mode=True, model_complexity=2)

# Lectura de la imagen de entrada y procesamiento con MediaPipe.
image = cv2.imread('./img_mp/img1.jpg')
results = pose.process(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

# Creación de una imagen en blanco para visualizar los puntos clave.
height, width, _ = image.shape
black_image = np.zeros((height, width, 3), np.uint8)

# Dibujo de los puntos clave y las conexiones en la imagen en blanco.
if results.pose_landmarks:
    mp.solutions.drawing_utils.draw_landmarks(
        black_image, results.pose_landmarks, mp_pose.POSE_CONNECTIONS)

cv2.imwrite('./img_mn/img2.jpg', black_image)

# Medición del tiempo de procesamiento.
```

```
elapsed_time = time.time() - start_time
print(f"Tiempo de ejecución: {elapsed_time} segundos.")
```

Este código resalta la facilidad con la que se puede cargar y procesar una imagen utilizando MediaPipe. El bloque crucial es el uso de 'pose.process' que toma una imagen RGB y devuelve los puntos clave detectados. Como se puede observar en la Figura 4.2, los puntos clave extraídos son precisos y bien distribuidos a lo largo del cuerpo humano, lo que permite una representación detallada del esqueleto en la imagen resultante. Esta precisión es fundamental para la generación efectiva de esqueletos representativos, que son esenciales para el análisis avanzado de posturas y movimientos en aplicaciones de visión por ordenador.

Figura 4.2: Esqueleto humano resultante tras la extracción de puntos clave con MediaPipe.

Posteriormente, estos puntos se dibujan en una imagen en negro utilizando draw_landmarks, lo que resulta en una representación clara y nítida del esqueleto. Como se ilustra en la Figura 4.3.

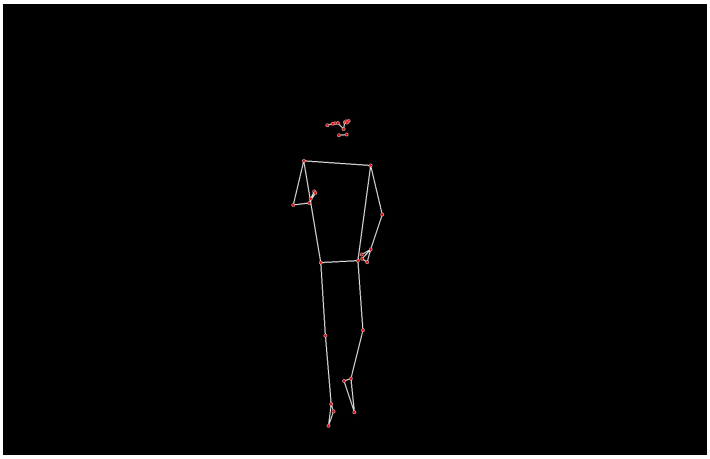

Figura 4.3: Salida del esqueleto humano tras la extracción de puntos clave con MediaPipe.

4.2.2.1 Análisis de Tiempos de Ejecución de MediaPipe

Para evaluar la eficiencia de MediaPipe, se registraron los tiempos de ejecución en múltiples instancias. La siguiente gráfica representa estos tiempos, facilitando una visualización directa de la consistencia y el rendimiento del programa.

Como se observa en la Figura 4.4, los tiempos de ejecución son consistentes, con una ligera variación que es normal en la ejecución de programas en entornos de sistemas operativos. Esto demuestra la fiabilidad y el alto rendimiento de MediaPipe para la detección de puntos clave en imágenes.

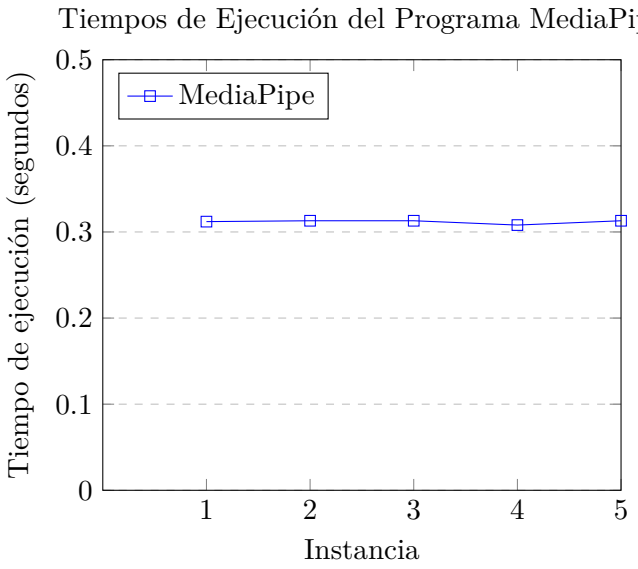

Figura 4.4: Tiempos de ejecución para cada instancia de la prueba con MediaPipe.

La eficiencia de MediaPipe en la extracción rápida y precisa de puntos clave la posiciona como una herramienta prometedora para su uso en etapas posteriores de este proyecto. No obstante, para tomar una decisión informada, se realizará una evaluación similar con otra librería de libre distribución, MoveNet, cuyos detalles y resultados se presentan a continuación.

4.2.3 MoveNet

La evaluación de las herramientas de visión por computador ha continuado con MoveNet, una red neuronal de vanguardia diseñada para la detección de poses humanas en tiempo real. El objetivo de estas pruebas es comparar la eficacia y eficiencia de MoveNet con respecto a MediaPipe, poniendo énfasis en la velocidad de procesamiento y la precisión en la extracción de puntos clave. Con tal fin, se ha utilizado la misma imagen que en el caso de MediaPipe para mantener una comparación coherente.

La detección de puntos clave utilizando MoveNet es un proceso complejo que implica varios pasos. Primero, se carga el modelo preentrenado de TensorFlow Hub. Este modelo está preentrenado con un conjunto de datos masivo de imágenes de personas con puntos clave

etiquetados. Una vez que el modelo está cargado, se utiliza para procesar la imagen de entrada. Este proceso implica convertir la imagen a un formato compatible con el modelo y luego aplicar el modelo a la imagen. Finalmente, los puntos clave detectados se visualizan en la imagen.

El código siguiente muestra el uso de MoveNet para detectar puntos clave. Un aspecto destacable es que, a diferencia de MediaPipe, MoveNet requiere que la imagen de destino sea cuadrada para dibujar correctamente los puntos clave. Esto se logra mediante un preprocesamiento que ajusta el tamaño de la imagen de entrada.

Código 4.2: Script de detección de puntos clave con MoveNet.

```
import tensorflow as tf
import tensorflow_hub as hub
import cv2
import time
import numpy as np

# Cargar el modelo MoveNet de TensorFlow Hub.
model = hub.load('https://tfhub.dev/google/movenet/singlepose/lightning/4')
movenet = model.signatures['serving_default']

# Marca de tiempo al inicio.
start_time = time.time()

# Cargar la imagen y preprocesarla.
image = tf.io.read_file('./img_mn/img1.jpg')
image = tf.io.decode_jpeg(image)
image = tf.image.resize_with_pad(image, 192, 192)
input_image = tf.cast(image, dtype=tf.int32)
input_image = tf.expand_dims(input_image, axis=0)

# Realizar inferencia.
results = movenet(input_image)

# Extraer los puntos clave del resultado de la inferencia.
keypoints_with_scores = results['output_0'].numpy()[0,0,:]

# Cargar la imagen original para dibujar los keypoints.
original_image = cv2.imread('./img_mn/img1.jpg')
output_image = original_image.copy()

# Dibujar los keypoints en la imagen.
for idx, keypoint in enumerate(keypoints_with_scores):
    if keypoint[2] > 0.5: # solo dibujar keypoints con confianza mayor a 0.5
        cv2.circle(output_image, (int(keypoint[1] * original_image.shape[1]), ↩
            ↪ int(keypoint[0] * original_image.shape[0])), 5, (0, 255, 0), -1)

cv2.imwrite('./img_mn/img2.jpg', output_image)

# Marca de tiempo al final y cálculo del tiempo de procesamiento.
```

```
end_time = time.time()
elapsed_time = end_time - start_time
print(f"Tiempo de ejecución: {elapsed_time} segundos.")
```

Este fragmento de código destaca cómo MoveNet se integra con TensorFlow Hub para facilitar la detección de poses. A diferencia de MediaPipe, MoveNet requiere de un preprocesamiento adicional para ajustar la imagen al tamaño de entrada del modelo, lo que puede influir en el tiempo total de ejecución y en la calidad de la imagen a procesar.

La figura 4.5 muestra la salida visual después de aplicar MoveNet a la imagen original. La imagen resultante ilustra los puntos clave identificados superpuestos sobre la imagen del sujeto.

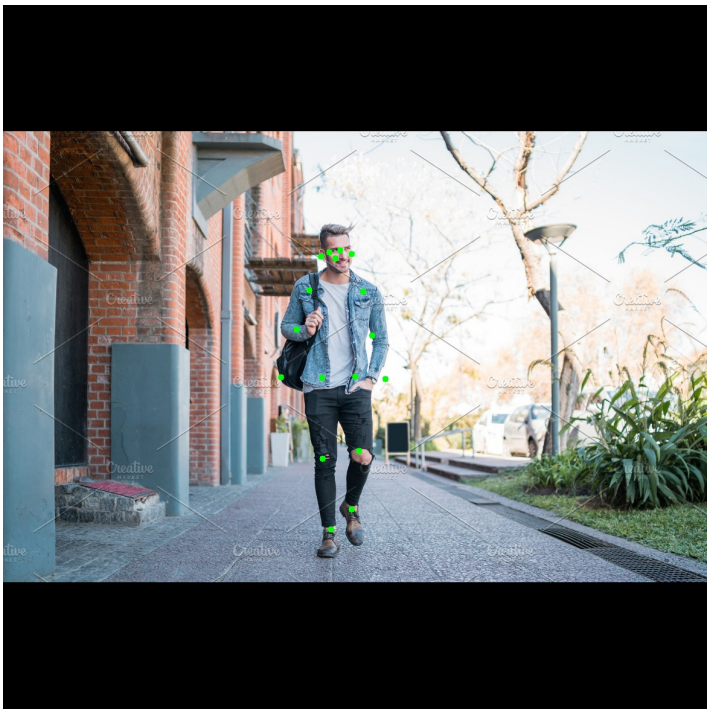

Figura 4.5: Resultados de la detección de puntos clave utilizando MoveNet.

4.2.3.1 Análisis de Tiempos de Ejecución de MoveNet

Al igual que con MediaPipe, se registraron los tiempos de ejecución para cada instancia de ejecución de MoveNet. Los datos recopilados proporcionan una medida cuantitativa de la eficiencia de MoveNet en la tarea de detección de poses.

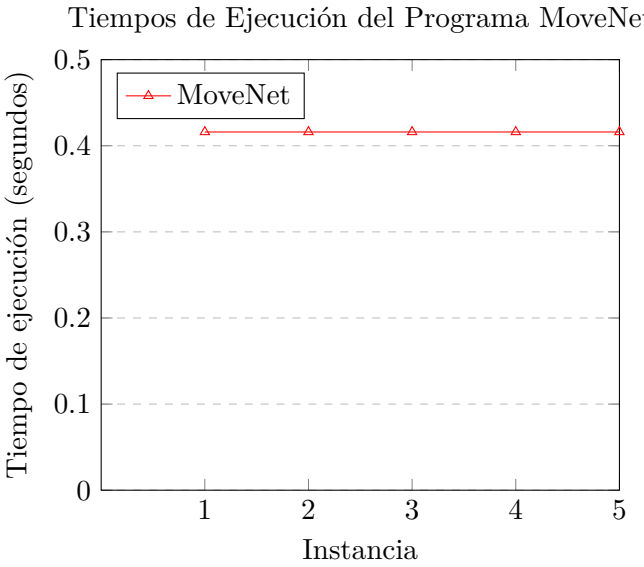

Figura 4.6: Tiempos de ejecución para cada instancia de la prueba con MoveNet.

La Figura 4.6 muestra que los tiempos de ejecución de MoveNet son ligeramente superiores en comparación con MediaPipe, lo cual puede atribuirse al proceso de preprocesamiento de la imagen. No obstante, la precisión y la facilidad de uso de MoveNet son factores que pueden compensar elaumento en el tiempo de procesamiento.

A continuación, se llevará a cabo un análisis comparativo para determinar cuál de las dos herramientas se ajusta mejor a los requisitos del proyecto en términos de eficiencia, precisión y usabilidad.

4.2.4 Análisis Comparativo entre MediaPipe y MoveNet

Tras evaluar tanto MediaPipe como MoveNet, es esencial realizar un análisis comparativo para determinar cuál de estas herramientas se adapta mejor a los objetivos del proyecto. Este análisis se basa en la calidad de la detección de puntos clave, los tiempos de ejecución y la idoneidad general para el proyecto.

En cuanto a la calidad de la detección de puntos clave, ambas herramientas han demostrado ser eficaces. MediaPipe y MoveNet han podido identificar con precisión los puntos clave del cuerpo humano en las mismas imágenes utilizadas en las pruebas, lo que permite una comparación directa de sus capacidades. Sin embargo, se observa que MediaPipe proporciona una representación ligeramente más precisa en términos de alineación y continuidad de los puntos clave en relación con la postura humana observada en la imagen original. Esto se refleja claramente en los resultados visuales presentados: como se puede observar en la Figura 4.2 para MediaPipe y la Figura 4.5 para MoveNet, los puntos clave detectados por MediaPipe se alinean más estrechamente con las posturas naturales humanas. Esta alineación proporciona una representación más coherente y continua, evidenciando la superioridad de MediaPipe en este aspecto específico del análisis de posturas.

Los tiempos de ejecución son un factor crucial, especialmente en aplicaciones en tiempo real. Así pues, se registraron los tiempos de ejecución para cada bilbioteca, como se muestra

en las Figuras 4.4 y 4.6, aunque una comparativa de los tiempos obtenidos para cada una de las imágenes puede observaser en la Figura 4.7.

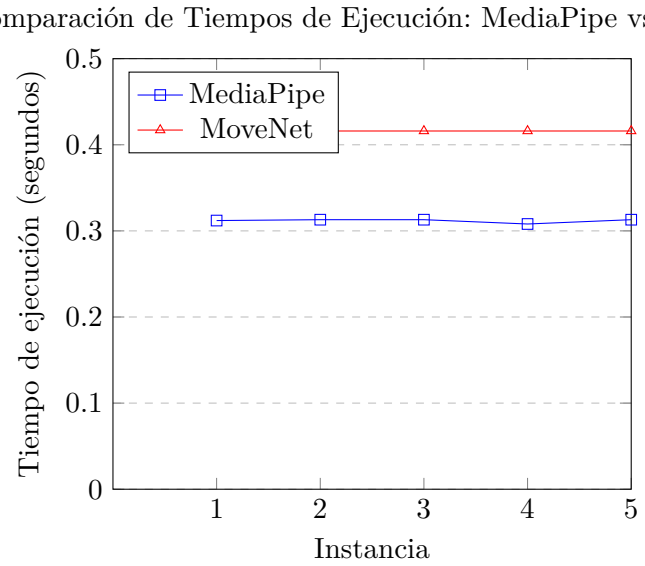

Figura 4.7: Comparación de los tiempos de ejecución entre MediaPipe y MoveNet.

Como se puede observar en la Figura 4.7, MediaPipe mostró consistentemente tiempos de ejecución más rápidos en comparación con MoveNet. Esta diferencia es significativa, puesto que se tiene como objetivo una aplicación de reconocimiento en tiempo real.

Considerando tanto la calidad de la detección de puntos clave como los tiempos de ejecución, MediaPipe se ha seleccionado como la herramienta a utilizar en este proyecto. Su capacidad para proporcionar detecciones precisas y rápidas lo hace más adecuado para el proyecto, debido al elevado número de imágenes a procesar y la precisión requerida para obtener un reconocimiento preciso y adecuado. Además, la facilidad de uso y la integración de MediaPipe en el ecosistema de Python ofrecen una ventaja adicional en términos de desarrollo y mantenimiento del proyecto.

4.3 Procesamiento de Datos

4.3.1 Entorno de Trabajo y Herramientas

En el desarrollo del proyecto, se ha implementado una estructura de trabajo rigurosa y disciplinada, esencial para la gestión eficiente del flujo de datos y la colaboración efectiva. La metodología se ha centrado en la utilización de un servidor remoto proporcionado por el departamento de Ciencia de la Computación e Inteligencia Artificial de la Universidad de Alicante, junto con herramientas colaborativas.

Para maximizar la eficiencia y comodidad en el desarrollo, se ha establecido una metodología que implica la redacción y depuración del código en un entorno local antes de su traslado y ejecución en el servidor remoto. Este enfoque ha permitido agilizar el proceso de desarrollo y garantizar la funcionalidad del código antes de su implementación. Para acceder al servidor remoto, se ha hecho uso del protocolo SSH.

En cuanto a la configuración del entorno de desarrollo, se ha utilizado Anaconda para crear y gestionar entornos virtuales. Esta plataforma ha permitido mantener configuraciones específicas para diferentes versiones de librerías, como TensorFlow 1.14 y 2.4, en entornos aislados. La funcionalidad de exportación de Conda ha facilitado la replicación exacta del entorno de desarrollo local en el servidor remoto, asegurando la coherencia en las pruebas y ejecuciones.

4.3.2 Extracción y Procesamiento de Frames de Vídeo

El proceso de extracción y procesamiento de los datos de cada vídeo es fundamental para la preparación del conjunto de datos necesario para entrenar el modelo de reconocimiento de acciones.

Antes de comenzar con las distintas etapas de procesamiento y análisis de los datos, es esencial mencionar la ubicación y la estructura de almacenamiento del dataset utilizado en este proyecto. Los datos del dataset, que consisten en una gran cantidad de vídeos relevantes para el reconocimiento de acciones, se encuentran almacenados en el servidor remoto, en la ruta `/home/joseburgos/datos`.

Esta ubicación en el servidor alberga todos los vídeos que forman parte del dataset Toyota Smarthome, previamente mencionado, y es desde donde se accede a estos para las sucesivas fases de extracción de frames, procesamiento y análisis. La elección de esta ruta de almacenamiento se debe a su capacidad para manejar grandes volúmenes de datos y la eficiencia en el acceso y procesamiento de los mismos dentro del entorno de trabajo del servidor remoto.

El primer paso en el procesamiento de los datos ha sido la extracción de frames individuales de los vídeos. Para esto, se ha implementado el script `extract_frames.py`. Este script utiliza la biblioteca OpenCV para leer y procesar vídeos, extrayendo cada frame y guardándolo como una imagen independiente. A continuación, se muestra un fragmento relevante del código:

Código 4.3: Extract frames from video script.

```
1 def extract_frames_from_video(video_path, output_folder):
2     # Crear carpeta de salida si no existe
3     if not os.path.exists(output_folder):
4         os.makedirs(output_folder)
5
```

```
    cap = cv2.VideoCapture(video_path)
    frame_count = 0

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        frame_filename = f"{output_folder}/frame_{frame_count:04d}.jpg"
        cv2.imwrite(frame_filename, frame)
        frame_count += 1

    cap.release()

# Ejemplo de uso
video_path = "/path/to/video.mp4"
output_folder = "/path/to/output/folder"
extract_frames_from_video(video_path, output_folder)
```

Este script se ejecutó para cada vídeo del conjunto de datos, resultando en una serie de imágenes almacenadas en la carpeta especificada. Por ejemplo, la Figura 4.8 muestra un frame extraído de un vídeo del dataset.

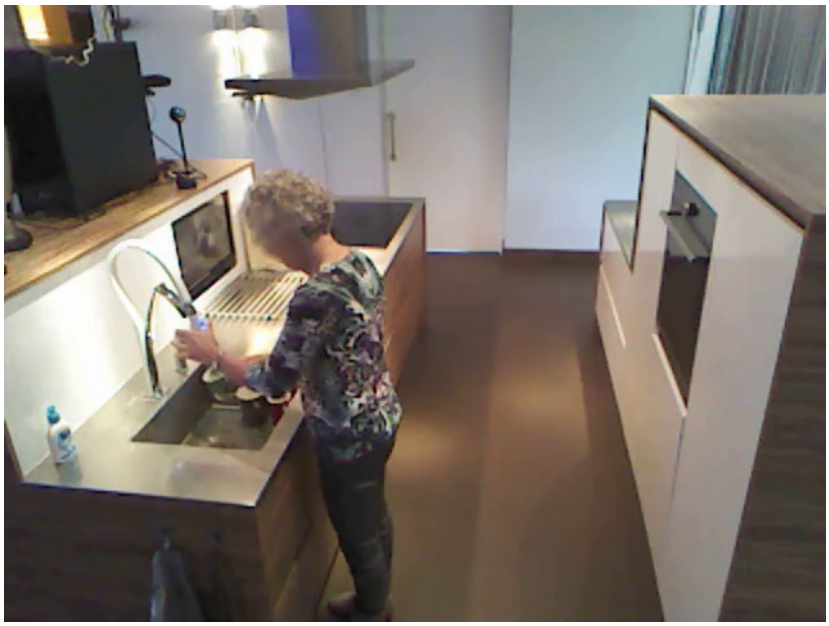

Figura 4.8: Ejemplo de un frame extraído de un vídeo.

Una vez extraídos los frames de los vídeos, se ha procedido a generar los esqueletos humanos y la extracción de keypoints. Este proceso se ha llevado a cabo utilizando el script `mediaPipe.py`, el cual aplica MediaPipe Pose a cada frame. MediaPipe Pose es una solución de visión por ordenador que permite la detección de la pose humana en tiempo real. El script procesa las imágenes para detectar la posición del cuerpo humano, generando un conjunto de

keypoints que representan las partes del cuerpo. El siguiente fragmento de código muestra cómo se implementó este proceso:

Código 4.4: Script para generación de esqueletos humanos y extracción de keypoints con MediaPipe.

```
mp_pose = mp.solutions.pose
pose = mp_pose.Pose(static_image_mode=True, model_complexity=2)

frame_folder = '/path/to/frames'
esqueleto_folder = '/path/to/esqueleto'
keypoints_folder = '/path/to/keypoints'

for frame_file in os.listdir(frame_folder):
    frame_path = os.path.join(frame_folder, frame_file)
    image = cv2.imread(frame_path)
    results = pose.process(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

    if results.pose_landmarks:
        # Generación y almacenamiento de la imagen del esqueleto
        esqueleto_image = np.zeros(image.shape, np.uint8)
        mp.solutions.drawing_utils.draw_landmarks(esqueleto_image, results.pose_landmarks, mp_pose.POSE_CONNECTIONS)
        esqueleto_path = os.path.join(esqueleto_folder, frame_file)
        cv2.imwrite(esqueleto_path, esqueleto_image)

        # Extracción y almacenamiento de keypoints en formato JSON
        keypoints_data = { "landmarks": [{ "x": lm.x, "y": lm.y, "z": lm.z, "visibility": lm.visibility } for lm in results.pose_landmarks.landmark] }
        keypoints_path = os.path.join(keypoints_folder, frame_file.replace('.jpg', '.json'))
        with open(keypoints_path, 'w') as f:
            json.dump(keypoints_data, f)

pose.close()
```

En este script, cada frame es procesado para generar una imagen que representa el esqueleto humano. Esta imagen se almacena en la carpeta especificada como esqueleto_folder. Posteriormente, los keypoints detectados se extraen y almacenan en formato JSON en keypoints_folder. La Figura 4.9 muestra un ejemplo de un esqueleto generado.

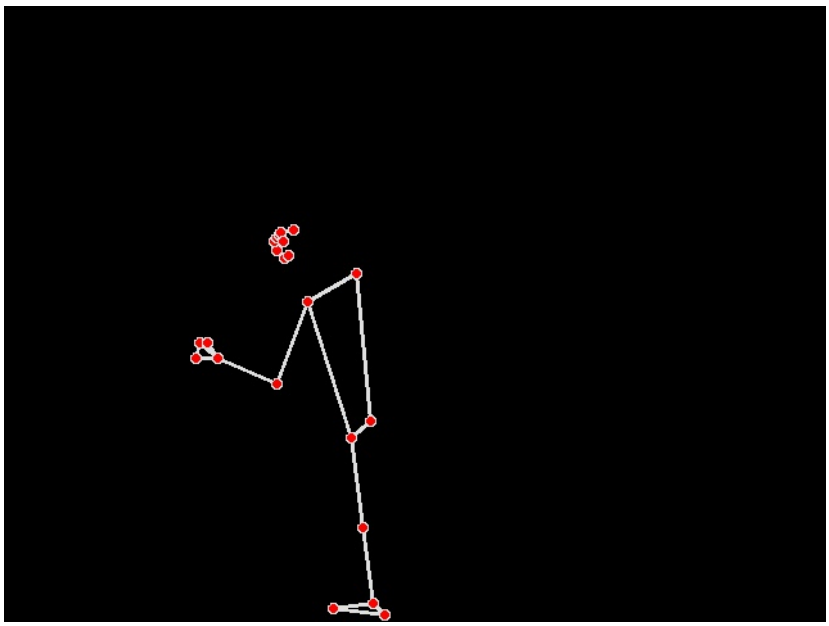

Figura 4.9: Ejemplo de un esqueleto humano generado a partir de un frame.

Los datos de keypoints se han almacenado en un formato JSON estructurado, proporcionando una representación numérica precisa de la posición (coordenadas x, y, z) y la visibilidad de cada punto clave. Un ejemplo de estos datos es el siguiente:

```
{
  "landmarks": [
    {
      "x": 0.33272433280944824,
      "y": 0.3966655433177948,
      "z": -0.06357371062040329,
      "visibility": 0.9999330043792725
    },
    ...
  ]
}
```

Este archivo JSON, como se muestra en el ejemplo, detalla la ubicación y visibilidad de puntos clave en un frame específico. Cada objeto dentro del arreglo landmarks representa un punto clave en el esqueleto humano detectado, donde x, y, y z son las coordenadas del punto en el espacio y visibility indica la probabilidad de que el punto clave sea visible en el frame.

Este enfoque integral para la extracción y procesamiento de datos aporta una base sólida para el análisis posterior y la implementación de modelos de aprendizaje profundo orientados al reconocimiento de acciones humanas.

4.3.3 Preparación de Frames para el Entrenamiento del Modelo

Una vez extraídos los frames de los vídeos y transformados en representaciones esqueléticas a través de la detección de puntos clave, el proceso de preparación de datos para el entrenamiento del modelo de aprendizaje profundo requiere una adaptación adicional de estos frames a un formato cuadrado. Esta necesidad surge debido a que numerosos modelos en el campo de la visión por ordenador muestran un rendimiento optimizado cuando se alimentan con imágenes de dimensiones uniformes.

Para explorar las técnicas disponibles de adaptación a formato cuadrado, se implementaron dos scripts específicos: uno para el recorte (*cropping*) y otro para el relleno (*padding*). A continuación, se presentan fragmentos de código significativos de cada script y se discuten las imágenes resultantes de aplicar cada técnica, ilustrando los efectos del recorte y el relleno en las representaciones esqueléticas de los frames.

4.3.3.1 Recorte de Frames (*Cropping*)

El script de recorte ajusta las dimensiones de los frames cortando secciones periféricas de la imagen para obtener un formato cuadrado. A continuación, se muestra un fragmento representativo del script de recorte:

Código 4.5: Fragmento del script de recorte (Cropping)

```
# Fragmento del script de recorte (Cropping)
def crop_to_square(image):
    height, width = image.shape[:2]
    new_size = min(height, width)
    start_x = (width - new_size) // 2
    start_y = (height - new_size) // 2
    return image[start_y:start_y+new_size, start_x:start_x+new_size]
```

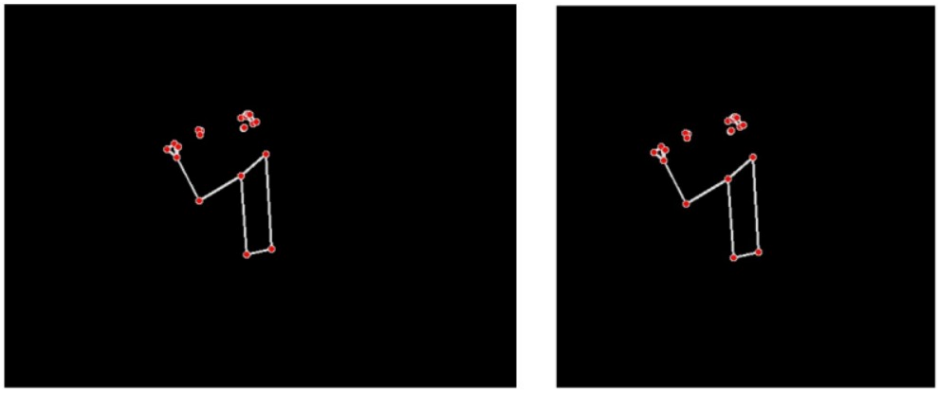

Figura 4.10: Comparación de un frame antes y después del recorte.

4.3.3.2 Relleno de Frames (*Padding*)

El *padding* o relleno de frames consiste en añadir márgenes de píxeles alrededor de las imágenes. Esta técnica se utiliza para aumentar las dimensiones de las imágenes hasta alcanzar un formato cuadrado, lo que permite mantener una uniformidad en el tamaño de las entradas a la red sin perder información visual importante. El relleno es especialmente útil para asegurar que todas las imágenes tengan las mismas dimensiones, facilitando el procesamiento por parte de las redes neuronales que requieren tamaños de entrada fijos. A continuación, se proporciona un fragmento representativo del script de relleno:

Código 4.6: Fragmento del script de relleno (Padding)

```
# Fragmento del script de relleno (Padding)
def pad_to_square(image):
    height, width = image.shape[:2]
    difference = abs(height - width)
    pad_side = difference // 2
    if height > width:
        return cv2.copyMakeBorder(image, 0, 0, pad_side, pad_side,
                                  cv2.BORDER_CONSTANT, value=[0, 0, 0])
    else:
        return cv2.copyMakeBorder(image, pad_side, pad_side, 0, 0,
                                  cv2.BORDER_CONSTANT, value=[0, 0, 0])
```

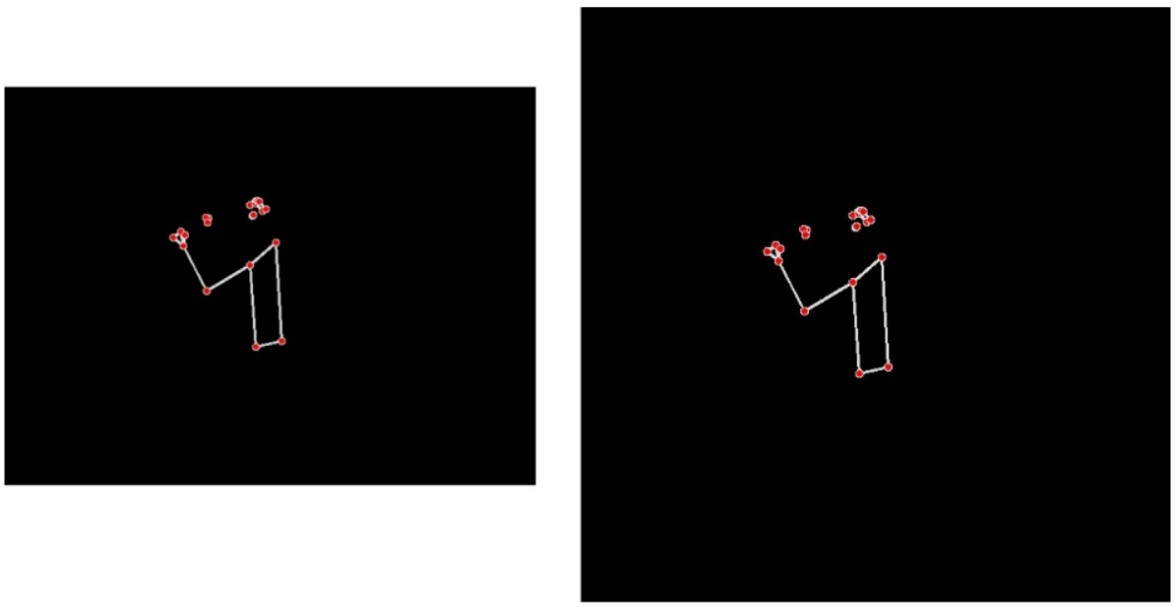

Figura 4.11: Comparación de un frame antes y después del relleno.

La evaluación comparativa de ambas técnicas ha sido crucial para determinar la idoneidad de cada una en el contexto de la preparación de datos para el modelo de reconocimiento de actividades. La preservación de la información esquelética completa, esencial para el apren-

dizaje efectivo del modelo, ha inclinado la preferencia hacia el uso del relleno para el ajuste de los frames. Este método asegura que la totalidad de los datos esqueléticos esenciales para el reconocimiento de actividades se mantiene intacta, sin sacrificar partes vitales del sujeto en la imagen.

4.3.4 Conversión de Frames de Esqueletos a Vídeos

Tras la preparación y adaptación de los frames a un formato cuadrado, el paso subsiguiente en la metodología consiste en la reconversión de estos frames de esqueletos en secuencias de vídeo. Este proceso es crucial para la generación de un conjunto de datos de entrada coherente y estructurado, adecuado para el entrenamiento del modelo de reconocimiento de actividades. Los vídeos resultantes, compuestos exclusivamente por representaciones esqueléticas, constituyen el input directo requerido por el modelo.

Para llevar a cabo esta conversión, ha sido implementado un script en Python que automatiza la unión de los frames individualmente procesados en secuencias de vídeo. Este script realiza una serie de operaciones esenciales, como el ordenamiento de los frames según su numeración, la creación de objetos *VideoWriter* para la composición del vídeo y, finalmente, la compilación de los frames en el formato de vídeo deseado. A continuación, se presenta un fragmento representativo del script utilizado:

Código 4.7: Fragmento simplificado del script de conversión de frames a vídeos.

```
def sort_frames(frame):
    # Extrae el número del nombre del archivo para el ordenamiento
    return int(re.search(r'\d+', frame).group())

def create_video_from_frames(frames_path, output_video_path, fps=20):
    frames = sorted(os.listdir(frames_path), key=sort_frames)
    first_frame = cv2.imread(os.path.join(frames_path, frames[0]))
    frame_size = (first_frame.shape[1], first_frame.shape[0])
    out = cv2.VideoWriter(output_video_path, cv2.VideoWriter_fourcc(*'mp4v'), ↩
        ↪ fps, frame_size)

    for frame_name in frames:
        frame = cv2.imread(os.path.join(frames_path, frame_name))
        out.write(frame)

    out.release()
```

Este procedimiento garantiza que los vídeos generados mantienen una consistencia temporal y espacial, respetando la secuencia original de las acciones capturadas. Cada vídeo resultante encapsula una serie continua de movimientos, facilitando así la tarea del modelo de aprender patrones de movimiento y reconocer actividades específicas.

4.3.5 Generación de Archivos CSV para la Organización de Datos

Con el fin de organizar de manera eficiente el conjunto de datos y facilitar su posterior uso en el entrenamiento y evaluación del modelo de reconocimiento de actividades, ha sido implementado un script en Python para generar archivos CSV. Estos archivos sirven como índices, enumerando los vídeos disponibles junto con sus correspondientes etiquetas de categoría de acción. La generación de estos índices en formato CSV permite una gestión más sencilla de los datos, además de habilitar la automatización de la carga de datos durante las fases de entrenamiento y validación del modelo.

El script automatiza el proceso de recorrido del directorio que contiene los vídeos de esqueletos, recopilando la ruta de cada vídeo y su etiqueta de categoría, basada en la estructura del directorio. Posteriormente, se emplea la biblioteca pandas para manipular esta información y dividirla en dos conjuntos: entrenamiento y prueba. Esta división se realiza de manera estratificada para asegurar que ambas particiones reflejen la distribución original de las categorías de acción en el conjunto de datos completo.

A continuación, se muestra un fragmento representativo del script utilizado:

Código 4.8: Fragmento del script para la generación de archivos CSV.

```
def generate_csv(root_dir, train_csv_path, test_csv_path, test_size=0.2):
    data = []
    for root, dirs, files in os.walk(root_dir):
        for file in files:
            if file.endswith('.mp4'):
                tag = os.path.basename(root)
                relative_path = os.path.relpath(os.path.join(root, file), ↩
                    ↪ root_dir)
                data.append({'video_name': relative_path, 'tag': tag})
    df = pd.DataFrame(data)
    train_df, test_df = train_test_split(df, test_size=test_size, stratify=df['↩
        ↪ tag'])
    train_df.to_csv(train_csv_path, index=False)
    test_df.to_csv(test_csv_path, index=False)
```

La generación de estos archivos CSV no solo optimiza el flujo de trabajo de preprocesamiento de datos sino que también garantiza que el proceso de entrenamiento y evaluación del modelo sea reproducible y fácilmente adaptable a nuevos conjuntos de datos. La estrategia de división de datos empleada apoya la validación cruzada y el análisis exhaustivo del rendimiento del modelo, constituyendo un paso crítico en la preparación de los datos para el aprendizaje profundo.

4.4 Modelos de Deep Learning Implementados

4.4.1 Modelo Basado en ConvLSTM para Reconocimiento de Actividades

En esta sección, se detalla el primer modelo implementado para el reconocimiento de actividades en secuencias de vídeo. Este modelo utiliza una arquitectura basada en ConvLSTM, la cual es particularmente adecuada para el procesamiento de datos secuenciales con una dimensión espacial, como es el caso de los frames de un vídeo.

Las Redes Neuronales Convolucionales (CNNs) son un tipo de red neuronal especializada en el procesamiento de datos que tienen una estructura de cuadrícula, como las imágenes. Las CNNs utilizan capas de convolución que aplican filtros para detectar características locales como bordes, texturas y patrones en diferentes regiones de una imagen. Estas características locales se combinan a través de capas de pooling y capas densas para realizar tareas de clasificación o detección. En el contexto del reconocimiento de actividades en vídeo, las CNNs se utilizan para extraer características espaciales de cada frame individual del vídeo.

Las redes Long Short-Term Memory (LSTM) son un tipo de red neuronal recurrente diseñada para procesar secuencias temporales, manteniendo información a través de estados ocultos que se actualizan en cada paso de tiempo. La combinación de CNNs y LSTMs en una capa ConvLSTM2D permite capturar tanto las características espaciales como las temporales de los cuadros de vídeo.

4.4.1.1 Arquitectura del Modelo

El modelo se construye utilizando TensorFlow y Keras, comenzando con la importación de las bibliotecas necesarias para el procesamiento de datos, la manipulación de archivos, y la construcción y entrenamiento de modelos de aprendizaje profundo. La arquitectura del modelo consta de varias capas ConvLSTM2D intercaladas con capas de BatchNormalization para normalizar los datos de entrada por batch y acelerar el entrenamiento. Un ”batch” se refiere a un subconjunto de datos del conjunto de entrenamiento total que se utiliza para realizar una única actualización de los parámetros del modelo durante el entrenamiento. Utilizar batches ayuda a hacer el proceso más eficiente y puede contribuir a una mejor convergencia del modelo. La red se completa con una capa Flatten para transformar los datos a un formato adecuado para las capas densas, seguida por una capa densa con activación ReLU y una capa de Dropout para reducir el sobreajuste. Finalmente, se utiliza una capa densa con activación softmax para la clasificación de las actividades en el número deseado de clases.

4.4.1.2 Esquema de la Arquitectura del Modelo

Para ilustrar visualmente la arquitectura del modelo descrito, a continuación se presenta un esquema básico:

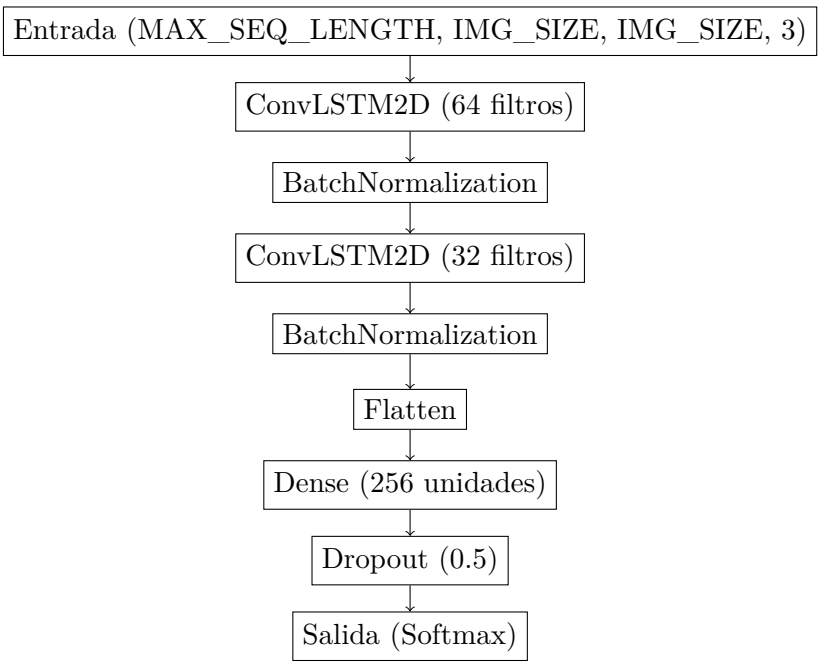

Figura 4.12: Esquema de la arquitectura del primer modelo de reconocimiento de actividades.

4.4.1.3 Descripción Detallada de las Capas del Modelo

Cada capa en la arquitectura del modelo ConvLSTM desempeña una función esencial en el procesamiento y clasificación de secuencias de vídeo. A continuación, se explica el propósito y configuración de cada capa de manera detallada:

- **Capa ConvLSTM2D:** Las capas ConvLSTM2D integran operaciones de convolución espacial (procesamiento de imagen) con Long Short-Term Memory (LSTM), que es una estructura de red neuronal diseñada para procesar secuencias temporales. Esta combinación permite capturar tanto las características espaciales como las temporales de los cuadros de vídeo. La primera capa utiliza 64 filtros con un tamaño de kernel de 3×3 y un ajuste de borde ('padding') tipo 'same', manteniendo la misma dimensión temporal entre las secuencias de entrada y salida. La segunda capa ConvLSTM2D utiliza 32 filtros, con las mismas dimensiones de kernel y ajuste de borde, pero sin retorno de secuencias, condensando la información temporal para su posterior clasificación.
- **Capa BatchNormalization:** Estas capas normalizan las activaciones de las capas anteriores, ajustando las escalas para estabilizar y acelerar el aprendizaje. La normalización por batch se realiza ajustando la media y varianza de las activaciones, lo que facilita el uso de tasas de aprendizaje más altas y reduce el número de épocas necesarias para entrenar el modelo efectivamente. Una "época" o "epoch", en el contexto de entrenamiento de modelos de machine learning, se define como un ciclo completo de entrenamiento en el cual el modelo es expuesto a la totalidad del conjunto de entrenamiento. Cada época consiste en una o más iteraciones, dependiendo del tamaño del batch y del conjunto de datos total, durante las cuales el modelo ajusta sus parámetros internos basándose en el error calculado tras cada batch.

- **Capa Flatten:** Esta capa transforma la salida tridimensional de la última capa ConvLSTM2D en un vector unidimensional. Este proceso es crucial para preparar los datos para la entrada a las capas densas, que no aceptan datos en formatos multidimensionales.
- **Capa Dense con activación ReLU:** Esta capa densa cuenta con 256 unidades y utiliza la función de activación ReLU (Rectified Linear Unit). La función ReLU introduce no linealidad en el modelo, permitiendo que este aprenda patrones más complejos en los datos, lo cual es vital para problemas de clasificación avanzados.
- **Capa Dropout:** Utiliza una tasa de dropout del 0.5 para mitigar el riesgo de sobreajuste. Esta técnica mejora la generalización del modelo al entrenamiento, ya que durante el mismo, omite aleatoriamente un porcentaje de las características aprendidas por la capa anterior en cada época.
- **Capa Dense con activación Softmax:** La última capa del modelo es una capa densa con activación softmax, diseñada para la clasificación en el número de clases especificadas por `NUM_CLASSES`. La función softmax transforma las salidas de la red en un vector de probabilidades, donde cada valor es la probabilidad de pertenecer a una de las clases, facilitando así la clasificación final.

4.4.1.4 Distribución de Datos para Entrenamiento, Validación y Tests

Antes de proceder al preprocesamiento y entrenamiento de los modelos, se ha realizado una distribución de los vídeos en conjuntos de entrenamiento, validación y pruebas. Para asegurar la validez del entrenamiento y la evaluación del modelo, es crucial mantener una representación equitativa de cada clase en los tres conjuntos. Se ha utilizado una proporción de 60/20/20 para dividir los vídeos, lo que significa que aproximadamente el 60% de los vídeos de cada clase se asignan al entrenamiento, el 20% al conjunto de validación y el 20% al conjunto de pruebas.

La siguiente tabla muestra la distribución exacta de vídeos por clase, incluyendo el total de vídeos, el número asignado al entrenamiento, validación y al conjunto de pruebas:

Clase	**Total**	**Entrenamiento**	**Validación**	**Tests**
Cleandishes	378	227	76	76
Cleanup	380	228	76	76
Cut	178	107	36	35
Stir	579	347	116	116
Usestove	96	58	19	19

Tabla 4.1: Distribución de vídeos por clase del primer modelo para los conjuntos de entrenamiento, validación y pruebas, basada en una división estratificada de 60/20/20.

Esta estrategia de división asegura que cada conjunto refleje fielmente la distribución de las clases en el conjunto de datos completo, lo cual es vital para prevenir el sesgo en el entrenamiento y validar la capacidad del modelo para generalizar a nuevos datos. Además, durante el entrenamiento, las muestras se seleccionan y se presentan al modelo de manera aleatoria, garantizando así que cada época del entrenamiento exponga al modelo a diferentes

combinaciones de datos. Esta aleatorización es crucial para prevenir el sobreajuste y asegurar que el modelo aprenda a generalizar a partir de los patrones en los datos en lugar de memorizar el orden específico de presentación.

4.4.1.5 Preprocesamiento de Datos y Entrenamiento

El preprocesamiento de los datos involucra la lectura y preparación de los frames que componen cada vídeo, utilizando las funciones de OpenCV para el manejo de vídeos y TensorFlow para el ajuste de tamaño y padding con relleno de los frames. Dependiendo de la prueba específica, los frames se redimensionan a diferentes tamaños para explorar el impacto de la resolución de la imagen en el rendimiento del modelo. Dado el volumen de datos, se utiliza una clase denominada `FrameGenerator` que facilita la división de los vídeos en frames y su preprocesamiento. Además, esta clase se encarga de cargar los frames en memoria de manera eficiente según sean necesarios para el entrenamiento, validación y prueba del modelo.

Para mejorar el rendimiento del entrenamiento del modelo, se emplean varias técnicas de manipulación de datos, tales como caching, shuffling y prefetching:

- **Caching:** Esta técnica almacena en caché el resultado de operaciones de preprocesamiento repetitivas y costosas. Al guardar estos datos en la memoria, se evita tener que recalcularlos en cada época, lo que acelera el proceso de entrenamiento.
- **Shuffling:** El mezclado aleatorio de los datos en cada época asegura que el modelo no vea los datos en el mismo orden cada vez. Esto ayuda a evitar patrones no deseados y mejora la capacidad del modelo para generalizar a partir de los datos de entrenamiento.
- **Prefetching:** Esta técnica permite que el preprocesamiento de los datos y el entrenamiento del modelo se realicen en paralelo. Mientras el modelo entrena con un batch de datos, el siguiente batch se carga y preprocesa en segundo plano, lo que reduce el tiempo total de entrenamiento.

Dependiendo de la prueba específica, los frames se redimensionan a diferentes tamaños para explorar el impacto de la resolución de la imagen en el rendimiento del modelo. El entrenamiento del modelo se realiza con un conjunto de datos configurado para el rendimiento mediante estas técnicas.

4.4.1.6 Test 1: EarlyStopping

En este test se ha utilizado un tamaño de imagen de 160x160 píxeles. El entrenamiento del modelo ConvLSTM se ha llevado a cabo en varias iteraciones con variaciones en el parámetro de patience de la función *EarlyStopping*, que se encarga de detener el entrenamiento cuando una métrica monitorizada deja de mejorar. Se han realizado un total de tres pruebas para evaluar la influencia de este parámetro en el rendimiento del modelo.

4.4.1.6.1 Prueba 1: EarlyStopping con patience=0 En la primera prueba se ha configurado el callback EarlyStopping con una paciencia de cero, deteniendo el entrenamiento inmediatamente después de que el valor de pérdida de validación ha aumentado por primera vez. Los resultados muestran que la arquitectura no es capaz de aprender el conjunto de datos

de entrenamiento, como se puede observar en la figura 4.13, donde la pérdida de validación aumenta significativamente tras pocas épocas, mientras que la pérdida de entrenamiento continúa disminuyendo. Similarmente, la figura 4.14 muestra que la precisión de validación no mejora, contrastando con la precisión de entrenamiento.

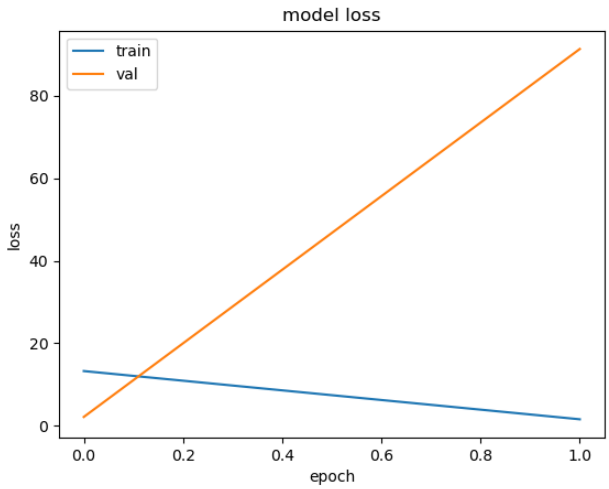

Figura 4.13: Pérdida del modelo con EarlyStopping configurado con patience=0.

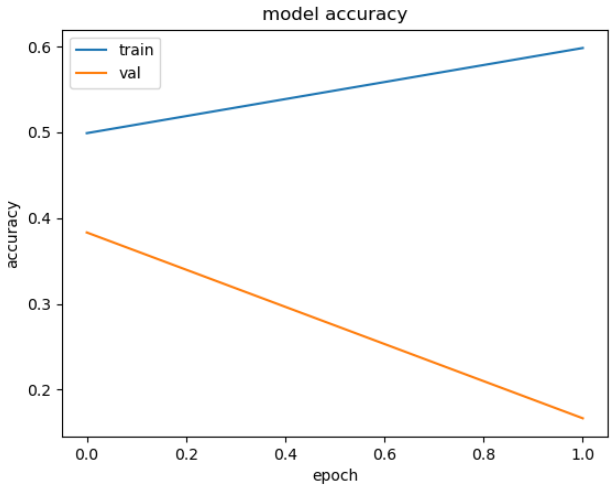

Figura 4.14: Precisión del modelo con EarlyStopping configurado con patience=0.

4.4.1.6.2 Prueba 2: EarlyStopping con patience=0.002 En la segunda prueba se ha ajustado la configuración de *patience* en la función de *EarlyStopping* a 0.002. Los resultados, presentados en las figuras 4.15 y 4.16, indican una mejora en la estabilización del entrenamiento, pero sin conseguir aprender correctamente los datos de entrenamiento. Aunque

la pérdida de validación sigue siendo alta, la precisión de validación muestra una mejora incremental.

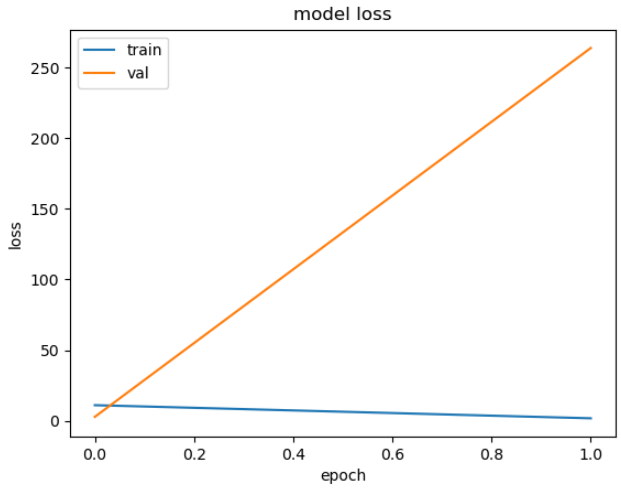

Figura 4.15: Pérdida del modelo con EarlyStopping configurado con patience=0002.

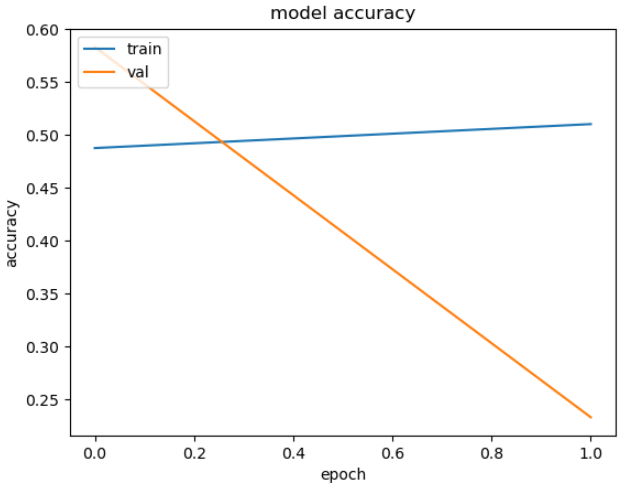

Figura 4.16: Precisión del modelo con EarlyStopping configurado con patience=0002.

4.4.1.6.3 Prueba 3: Sin EarlyStopping La tercera prueba se ha realizado sin la función EarlyStopping para permitir que el modelo completara todas las épocas planificadas. Las figuras 4.17 y 4.18 muestran los resultados de esta configuración. La ausencia de EarlyStopping ha permitido que el modelo continúe aprendiendo más allá del punto en que la primera mejora ha dejado de ser evidente. Se observa una pérdida de validación fluctuante y una precisión que un tanto más estable que en las pruebas anteriores, aunque en este caso se tiene

sobreajuste.

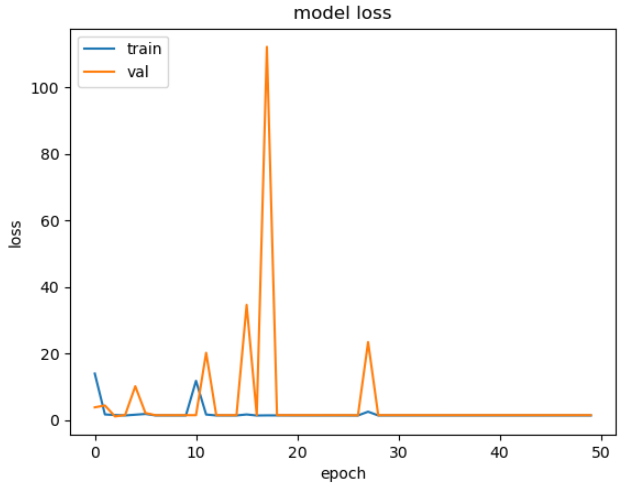

Figura 4.17: Pérdida del modelo sin la función EarlyStopping.

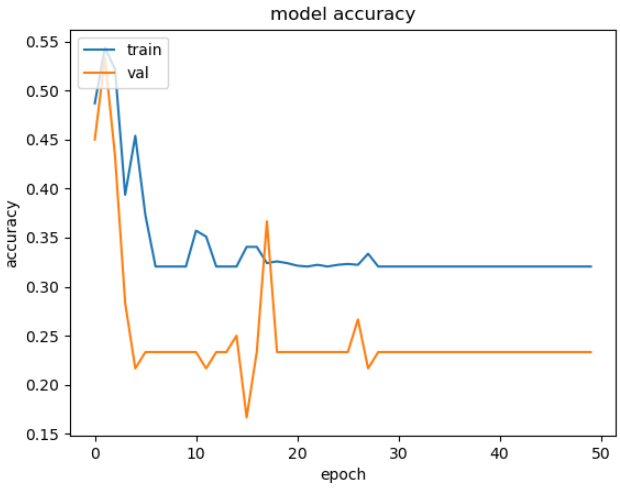

Figura 4.18: Precisión del modelo sin la función EarlyStopping.

4.4.1.7 Test 2: Modificación de Hiperparámetros y Arquitectura

Para la cuarta evaluación, se han sido introducido algunas modificaciones en la configuración de hiperparámetros del modelo, con un tamaño de imagen de entrada de 160x160 y un tamaño de batch de 2. Además, se ha ajustado el número de épocas a 25 y se ha cambiado la estructura del modelo ConvLSTM para tener capas con una cantidad menor de filtros y unidades en la capa densa. A continuación se presenta el análisis de los resultados obtenidos con esta configuración.

Durante el entrenamiento del modelo, se observa una mejora en la precisión de entrenamiento y validación con respecto a la primera prueba, como se ilustra en la Figura 4.19. A pesar de que la precisión de validación no alcanza niveles altos, su estabilidad a lo largo de las épocas sugiere que el modelo ha mejorado en términos de generalización.

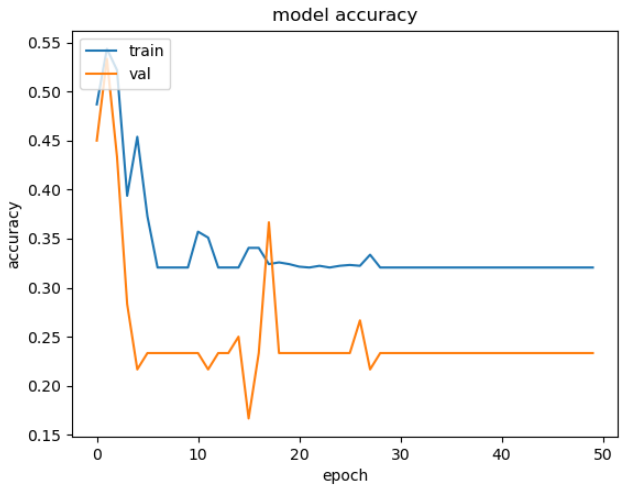

Figura 4.19: Precisión del modelo en la segunda evaluación.

La pérdida del modelo, presentada en la Figura 4.20, muestra un descenso estable en las primeras épocas seguido de una estabilización, lo que indica una convergencia adecuada del entrenamiento sin signos claros de sobreajuste.

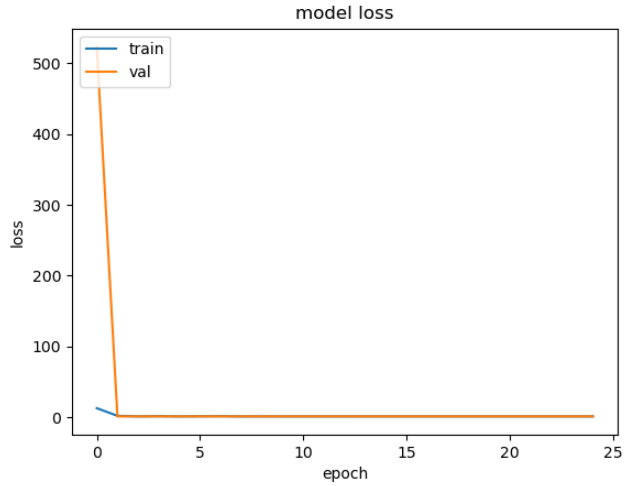

Figura 4.20: Pérdida del modelo en la segunda evaluación.

Al analizar la matriz de confusión mostrada en la Figura 4.21, se evidencia que el modelo ha exhibido una habilidad destacada para precisar correctamente la categoría *Stir*, logrando un total de 96 aciertos. No obstante, se detecta un sesgo significativo hacia esta clase, lo que resulta en la clasificación errónea de todas las demás categorías como *Stir*. Esto señala una importante área de mejora en cuanto a la capacidad del modelo para discernir entre las distintas clases.

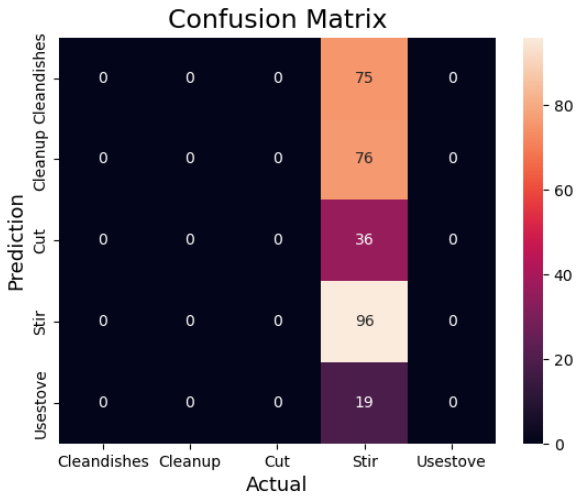

Figura 4.21: Matriz de confusión del modelo en la segunda evaluación.

4.4.1.8 Test 3: Evaluación con Variación del Tamaño de Batch y 25 Épocas

En el Test 3 se han explorado los efectos de diferentes tamaños de batch en el entrenamiento del modelo ConvLSTM. Durante el entrenamiento, los datos se agrupan en batches, permitiendo que el modelo ajuste sus parámetros internos progresivamente. Este test se ha centrado en evaluar cómo los distintos tamaños de batch afectan el rendimiento del modelo. Se han realizado pruebas con tamaños de batch de 4, 8 y 16, cada uno durante 25 épocas, con un tamaño de imagen de entrada de 160x160.

4.4.1.8.1 Prueba 1: Batch Size de 4 La primera prueba de este test se ha realizado con un tamaño de batch de 4. La precisión del modelo durante el entrenamiento muestra una tendencia ascendente, aunque con una variabilidad notoria en la validación, como se observa en la Figura 4.22. Esto puede sugerir que el modelo experimenta una cierta variabilidad en su capacidad de generalización a nuevos datos.

La pérdida del modelo, como se muestra en la Figura 4.23, indica una disminución consistente durante el entrenamiento, con picos esporádicos en la validación que podrían reflejar el impacto del tamaño de batch elegido en la estabilidad del entrenamiento. Asimismo, el hecho de que la validación aparezca sobre el entrenamiento pone de manifiesto un ligero sobreajuste del entrenamiento.

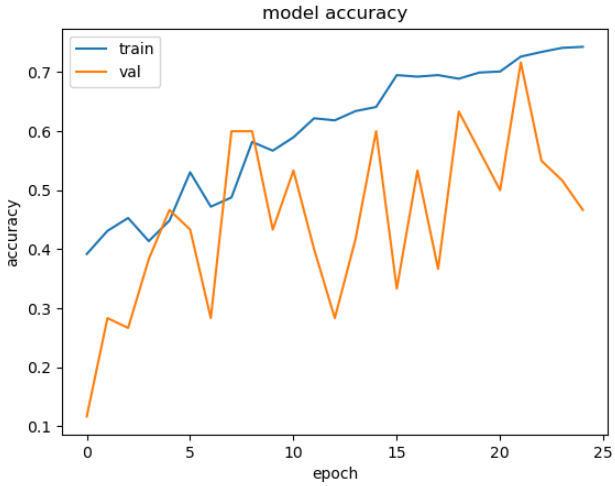

Figura 4.22: Precisión del modelo con un tamaño de batch de 4.

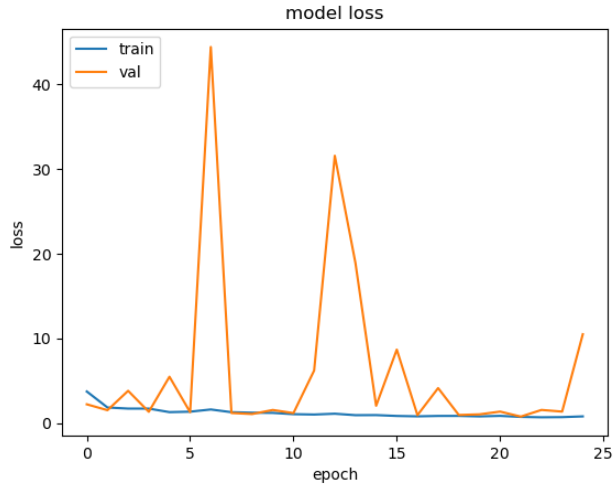

Figura 4.23: Pérdida del modelo con un tamaño de batch de 4.

La matriz de confusión, representada en la Figura 4.24, ofrece una visión más detallada del desempeño de clasificación del modelo. Aunque algunas clases se identifican con precisión, hay cierta confusión entre categorías haciendo necesario reajustar parámetros para mejorar los resultados en futuros entrenamientos.

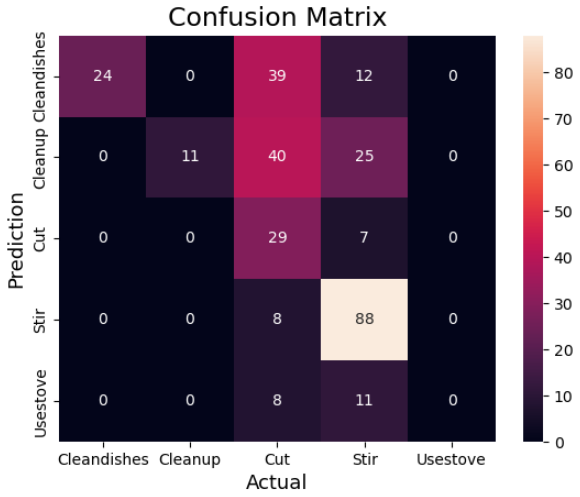

Figura 4.24: Matriz de confusión del modelo con un tamaño de batch de 4.

4.4.1.8.2 Prueba 2: Batch Size de 8 La segunda prueba del Test 3 se ha realizado con un tamaño de batch de 8. Observando la evolución de laprecisión del modelo (Figura 4.25), se puede apreciar una curva de aprendizaje más suavizada comparada con la prueba de batch de 4. La precisión de validación presenta menos fluctuaciones, lo que sugiere una mejor generalización del modelo cuando se incrementa el tamaño del batch.

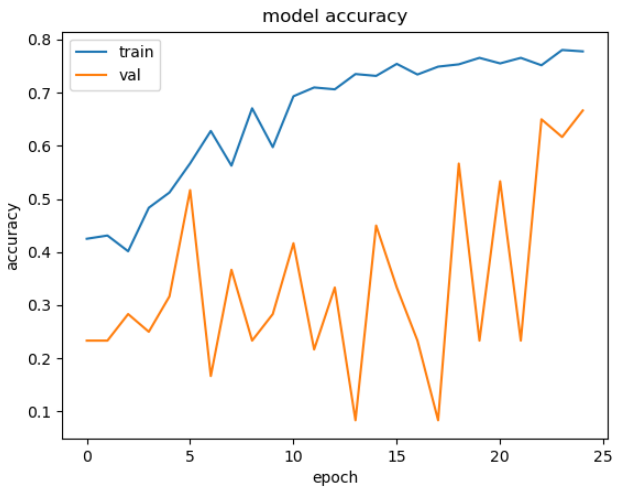

Figura 4.25: Precisión del modelo con un tamaño de batch de 8.

En términos de pérdida, la Figura 4.26 muestra un descenso progresivo en la pérdida de entrenamiento, mientras que la pérdida de validación se mantiene estable después de las primeras épocas.

La matriz de confusión para esta prueba, mostrada en la Figura 4.27, revela cómo el

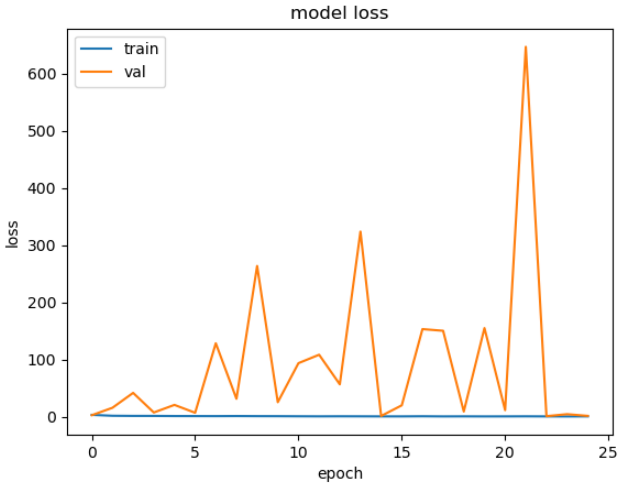

Figura 4.26: Pérdida del modelo con un tamaño de batch de 8.

modelo clasifica las diferentes categorías con un tamaño de batch de 8. Se puede observar una distribución de errores entre clases similar a la prueba con un batch de 4. Sin embargo, se notan ciertas mejoras en la clasificación de algunas categorías específicas, como por ejemplo, *Stir* y *Cleanup*. En particular, la categoría *Stir* muestra una reducción notable en la cantidad de falsos negativos, mientras que la categoría *Cleanup* presenta una mejora en la precisión, con menos casos clasificados erróneamente como *Cleandishes* en comparación con la configuración de batch anterior.

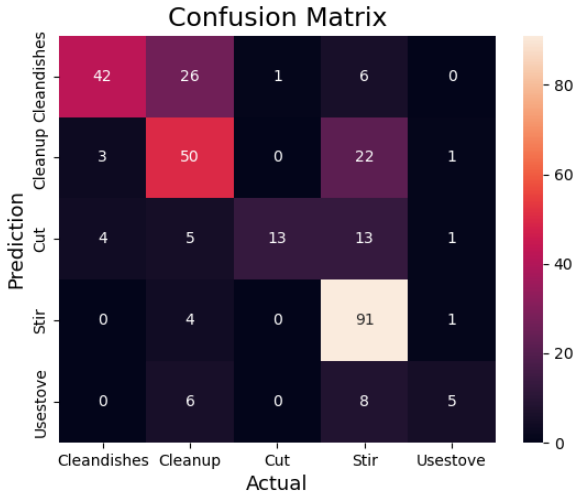

Figura 4.27: Matriz de confusión del modelo con un tamaño de batch de 8.

4.4.1.8.3 Prueba 3: Batch Size de 16 Para la tercera y última prueba del Test 3, se ha incrementado el tamaño de batch a 16. La Figura 4.28 muestra la curva de precisión

del modelo, donde se observa una precisión de entrenamiento alta y estable después de las primeras épocas. Sin embargo, la precisión de validación sugiere una tendencia a la baja, lo que podría indicar un sobreajuste a pesar de la alta precisión en los datos de entrenamiento.

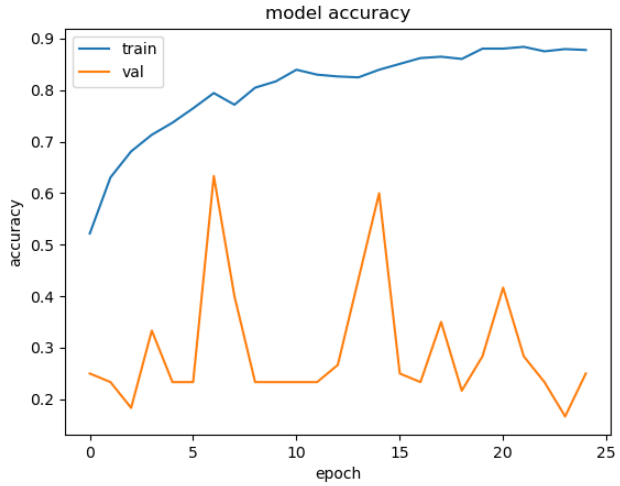

Figura 4.28: Precisión del modelo con un tamaño de batch de 16.

La pérdida del modelo, como se muestra en la Figura 4.29, presenta una menor variabilidad en la pérdida de entrenamiento y una tendencia general a la baja en la pérdida de validación, aunque sobre la de entrenamiento.

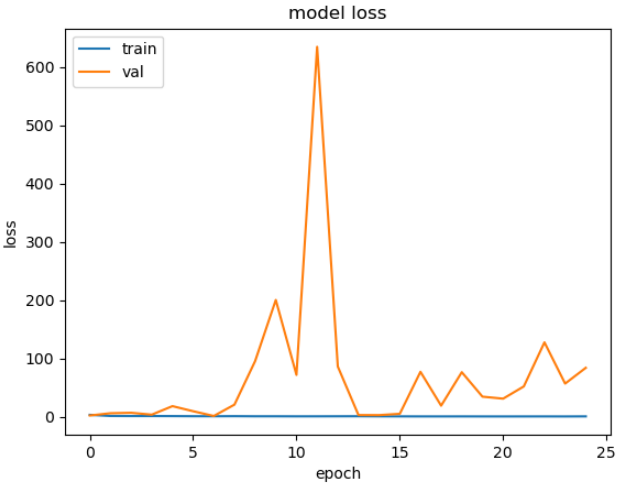

Figura 4.29: Pérdida del modelo con un tamaño de batch de 16.

Finalmente, la matriz de confusión para la prueba con batch de 16 se ilustra en la Figura 4.30. En esta figura, se aprecian los resultados de clasificación por categorías, donde se observa

que, nuevamente, todas las muestras de test se clasifican como una sola clase, *Stir*.

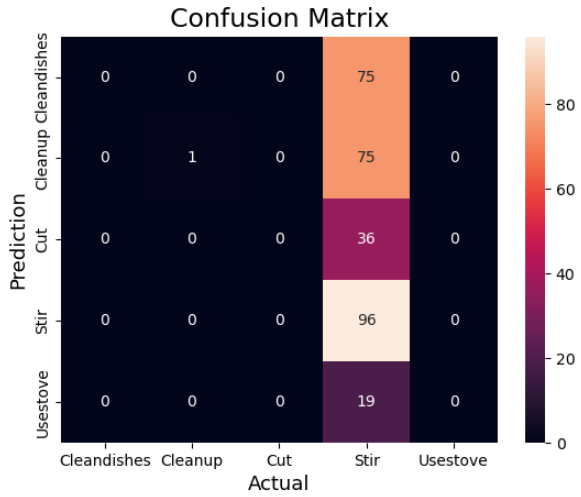

Figura 4.30: Matriz de confusión del modelo con un tamaño de batch de 16.

4.4.1.9 Test 4: Prueba con Batch Size de 16 y 50 Épocas

En este test se ha investigado el comportamiento del modelo ConvLSTM con un tamaño de batch de 16 a lo largo de 50 épocas. Este enfoque busca comprender la evolución del modelo en un proceso de entrenamiento más extenso.

Con un tamaño de batch de 16 y extendiendo el entrenamiento a 50 épocas, se observan cambios significativos en el comportamiento del modelo. La Figura 4.31 muestra que la precisión de entrenamiento se mantiene bastante estable y alta a lo largo de todas las épocas, mientras que la precisión de validación, aunque mejora, presenta altibajos que sugieren una cierta volatilidad en la generalización del modelo.

La pérdida del modelo, ilustrada en la Figura 4.32, refleja una disminución sostenida en la pérdida de entrenamiento y picos ocasionales en la pérdida de validación. Este comportamiento puede ser indicativo de puntos de ajuste potencial en el entrenamiento o en los datos de validación.

Finalmente, se examina la matriz de confusión en la Figura 4.33. Los resultados muestran una tendencia del modelo a clasificar correctamente la mayoría de las muestras de las clases *Cleandishes* y *Stir*. Sin embargo, el modelo tiene dificultades con las clases *Cleanup*, *Cut* y especialmente *Usetove*, donde la mayoría de las muestras han sido incorrectamente clasificadas como *Cleandishes*. Además, se observa que hay una considerable cantidad de muestras de *Cleandishes* que han sido erróneamente clasificadas como *Stir*.

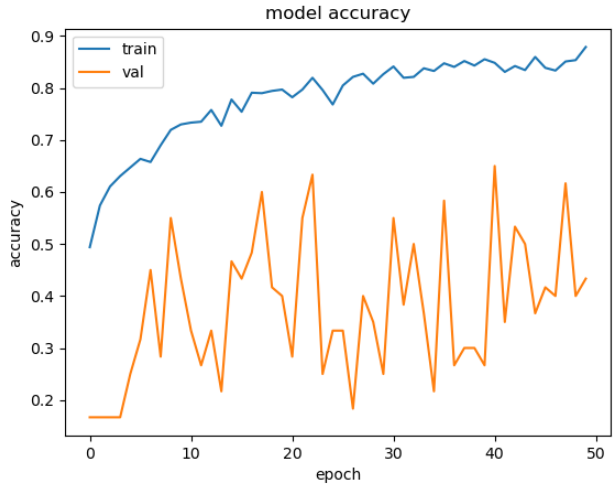

Figura 4.31: Precisión del modelo con un tamaño de batch de 16 durante 50 épocas.

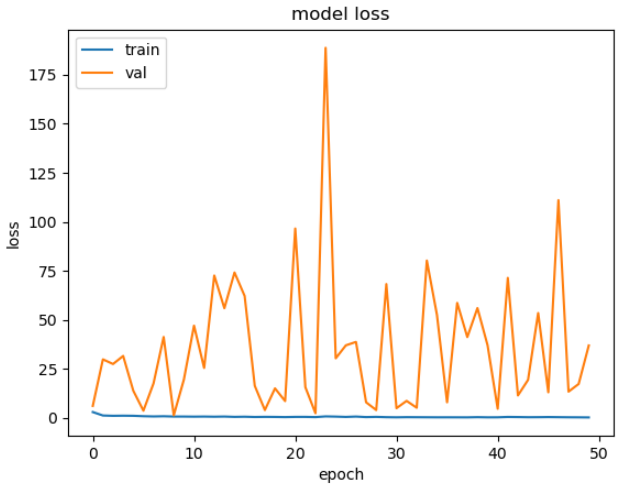

Figura 4.32: Pérdida del modelo con un tamaño de batch de 16 durante 50 épocas.

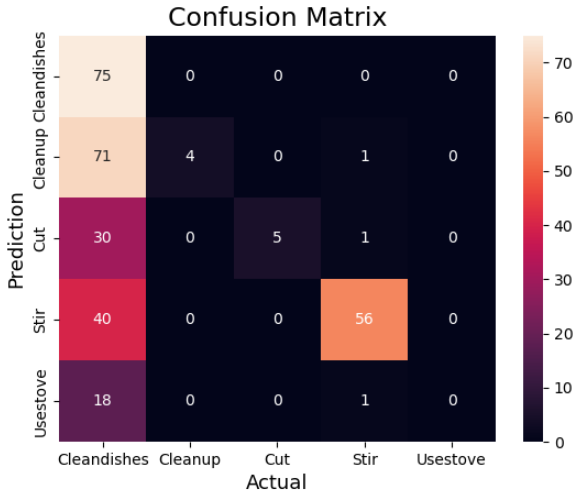

Figura 4.33: Matriz de confusión del modelo con un tamaño de batch de 16 durante 50 épocas.

4.4.1.10 Test 5: Evaluación con Tamaño de Imagen de 200x200

En este test se ha investigado el impacto de un aumento en la resolución de la imagen a 200x200 píxeles en el rendimiento del modelo ConvLSTM. Se han realizado dos pruebas con distintos tamaños de batch y número de épocas para analizar la evolución y capacidad de generalización del modelo bajo estas nuevas condiciones.

4.4.1.10.1 Prueba 1: 25 Épocas y Batch Size de 8 En la primera prueba de este test, con un tamaño de batch de 8 y durante 25 épocas, los resultados indican una tendencia general de mejora en el rendimiento del modelo. La Figura 4.34 muestra la curva de pérdida del modelo, donde se observa una rápida disminución de la pérdida en las primeras épocas, seguida por una estabilización a medida que el entrenamiento avanza, lo cual es indicativo de una buena convergencia del modelo.

Sin embargo, la Figura 4.35 presenta la precisión del modelo, donde se observa que la precisión de entrenamiento permanece relativamente estable a lo largo de las épocas, pero la precisión de validación es significativamente más baja. Esta discrepancia sugiere que el modelo podría estar sobreajustando los datos de entrenamiento y no generalizando adecuadamente a los datos no vistos.

La matriz de confusión, ilustrada en la Figura 4.36, muestra un patrón de clasificación donde la mayoría de las clases han sido incorrectamente identificadas como *Stir*. Sin embargo, la clase *Stir* ha sido en gran parte correctamente reconocida por el modelo. Este resultado sugiere que el modelo tiene una fuerte tendencia a predecir la clase *Stir*.

4.4.1.10.2 Prueba 2: 50 Épocas y Batch Size de 4 Esta prueba se ha extendido a 50 épocas con un tamaño de batch de 4 para evaluar la capacidad del modelo de adaptarse a un proceso de aprendizaje más largo con batches más pequeños. La Figura 4.37 muestra la precisión del modelo, donde se aprecia una estabilidad en la precisión de entrenamiento. No

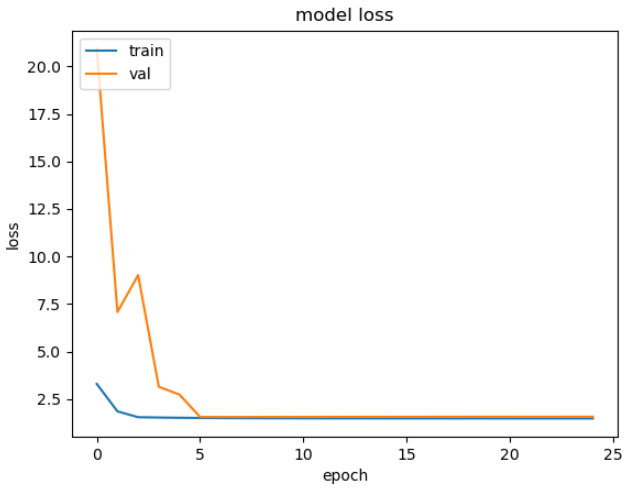

Figura 4.34: Pérdida del modelo con un tamaño de imagen de 200x200, 25 épocas y batch size de 8.

obstante, la precisión de validación experimenta fluctuaciones significativas, lo que podría indicar que el modelo tiene dificultades para generalizar a datos no vistos.

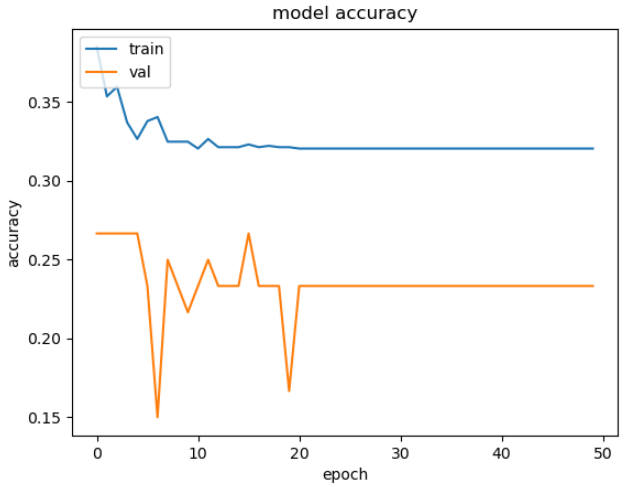

Figura 4.37: Precisión del modelo con un tamaño de imagen de 200x200, 50 épocas y batch size de 4.

La Figura 4.38 revela una curva de pérdida con picos elevados en la validación, lo que sugiere una gran variabilidad en el aprendizaje del modelo durante el proceso de entrenamiento. Estos picos pueden ser el resultado de un tamaño de batch demasiado pequeño, lo que a menudo puede llevar a una actualización de pesos menos estable.

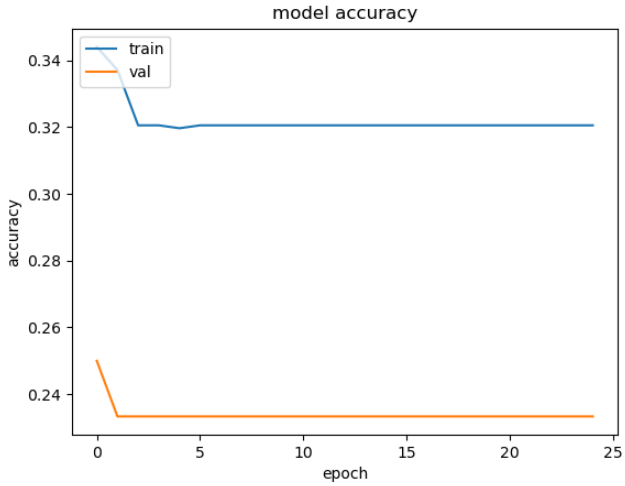

Figura 4.35: Precisión del modelo con un tamaño de imagen de 200x200, 25 épocas y batch size de 8.

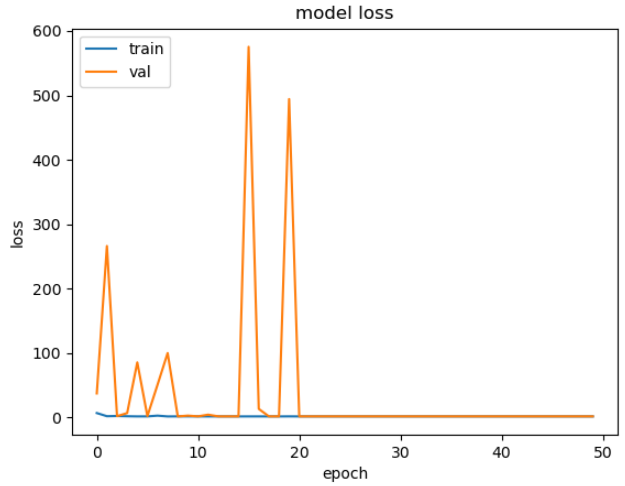

Figura 4.38: Pérdida del modelo con un tamaño de imagen de 200x200, 50 épocas y batch size de 4.

Al examinar la matriz de confusión en la Figura 4.39, se observa que el modelo ha demostrado una capacidad notable para identificar correctamente la clase *Stir* con un total de 96 aciertos. Sin embargo, presenta un sesgo significativo hacia esta clase, resultando en la clasificación incorrecta de todas las otras categorías como *Stir*, indicando un área de mejora considerable en el aprendizaje discriminativo entre las clases.

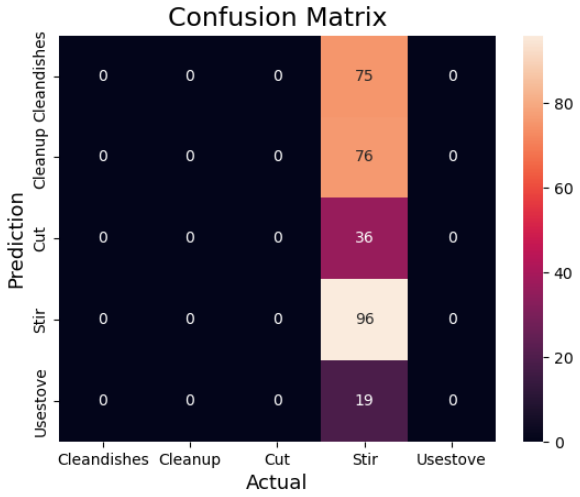

Figura 4.36: Matriz de confusión del modelo con un tamaño de imagen de 200x200, 25 épocas y batch size de 8.

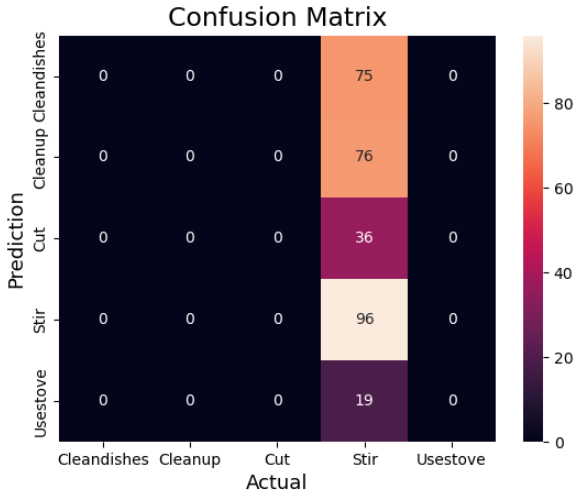

Figura 4.39: Matriz de confusión del modelo con un tamaño de imagen de 200x200, 50 épocas y batch size de 4.

4.4.1.11 Test 6: Evaluación con Tamaño de Imagen de 220x220

En este quinto test, se ha incrementado aún más la resolución de la imagen pasando a ser de 220x220 píxelescon el fin de analizar cómo responde el modelo ConvLSTM a imágenes de mayor resolución. Se plantean dos pruebas con el mismo tamaño de batch, pero diferentes números de épocas para investigar cómo el modelo gestiona un conjunto de entrenamiento visualmente más detallado a través de un proceso de entrenamiento de diferente duración.

4.4.1.11.1 Prueba 1: 25 Épocas y Batch Size de 8 Esta primera prueba, que se realiza con un tamaño de batch de 8 y durante 25 épocas, tiene como objetivo explorar la respuesta inicial del modelo al incremento en la resolución de las imágenes. La Figura 4.40 muestra la curva de precisión del modelo, revelando una estabilidad en la precisión de entrenamiento tras una caída inicial. Sin embargo, la precisión de validación exhibe variabilidad y mantiene una tendencia general por debajo de la precisión de entrenamiento, lo cual puede indicar la presencia de sobreajuste.

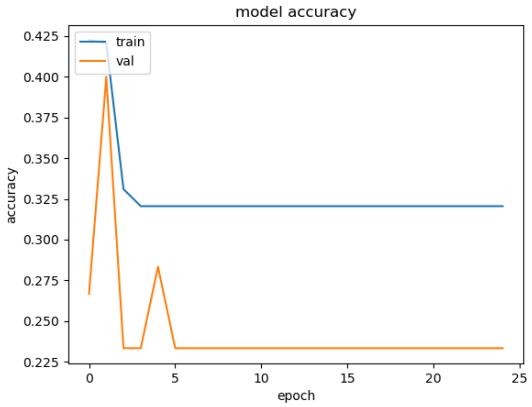

Figura 4.40: Precisión del modelo con un tamaño de imagen de 220x220, 25 épocas y batch size de 8.

La pérdida del modelo, que se muestra en la Figura 4.41, indica una rápida disminución en la pérdida de entrenamiento durante las primeras épocas, seguida de una estabilización. Aunque hay un pico notable en la pérdida de validación, este no se repite, lo que puede indicar un ajuste efectivo del modelo tras el incidente.

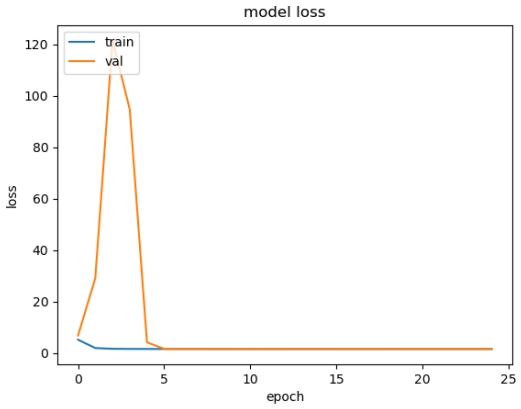

Figura 4.41: Pérdida del modelo con un tamaño de imagen de 220x220, 25 épocas y batch size de 8.

Sin embargo, al observar la matriz de confusión en la Figura 4.42, se pone de manifiesto que

el modelo muestra una fuerte tendencia a clasificar incorrectamente las muestras de las clases *Cleanup*, *Cut*, *Cleandishes* y *Usetove* como *Stir*. No obstante, la clase *Stir* se ha reconocido con mayor precisión.

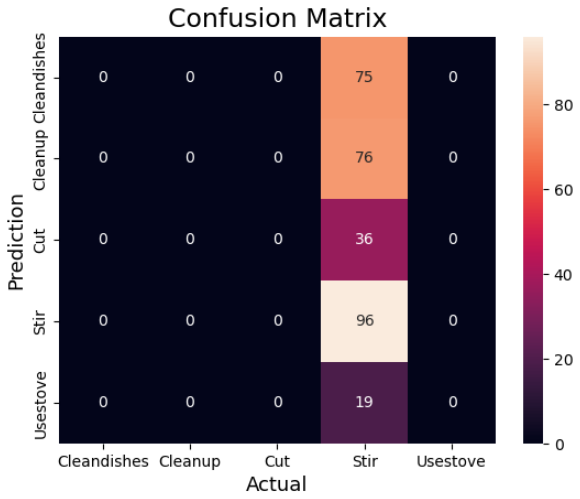

Figura 4.42: Matriz de confusión del modelo con un tamaño de imagen de 220x220, 25 épocas y batch size de 8.

4.4.1.11.2 Prueba 2: 50 Épocas y Batch Size de 8 Con la intención de profundizar en la evaluación del modelo, la segunda prueba del Test 5 se ha llevado a cabo con un tamaño de batch de 8 a lo largo de 50 épocas. Esta prueba busca examinar la consistencia del aprendizaje del modelo y su capacidad de generalización a lo largo de un proceso de entrenamiento más extendido.

La precisión del modelo, mostrada en la Figura 4.43, sugiere que el entrenamiento más prolongado no necesariamente conduce a un mejor resultado de generalización. La precisión de validación fluctúa considerablemente, mientras que la precisión de entrenamiento muestra una tendencia a estabilizarse después de las variaciones iniciales. Además, esta precisión disminuye durante el entrenamiento en lugar de mejorar. Este comportamiento puede ser indicativo de una necesidad de regularización más fuerte o de una revisión de la arquitectura del modelo para adaptarse mejor a la resolución incrementada.

La curva de pérdida, presentada en la Figura 4.44, exhibe una disminución significativa después de un pico inicial en las primeras épocas, seguida de una estabilización. Los picos en la pérdida de validación señalan episodios de aprendizaje inestable que requieren atención en el ajuste de hiperparámetros o en la mejora del proceso de validación.

Finalmente, la matriz de confusión (Figura 4.45) revela que, aunque el modelo muestra una capacidad notable para identificar correctamente la clase *Stir* con 96 aciertos, exhibe un sesgo considerable hacia clasificar erróneamente las muestras de las clases *Cleanup*, *Cut*, *Cleandishes*, y *Usetove* como *Stir*. Este patrón destaca una limitación significativa en el aprendizaje del modelo, incapaz de diferenciar adecuadamente entre varias clases y mostrando una tendencia a favor de *Stir*.

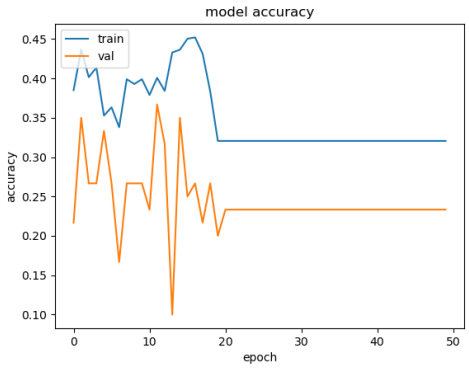

Figura 4.43: Precisión del modelo con un tamaño de imagen de 220x220 y 50 épocas con batch size de 8.

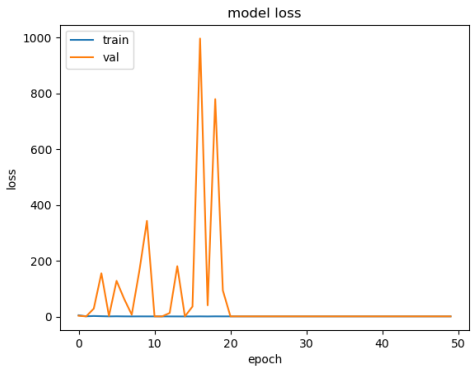

Figura 4.44: Pérdida del modelo con un tamaño de imagen de 220x220 y 50 épocas con batch size de 8.

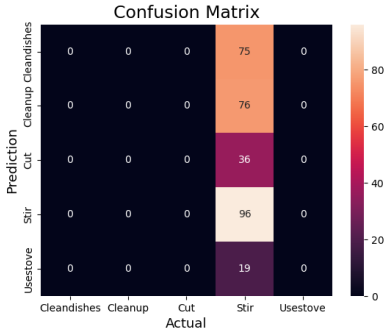

Figura 4.45: Matriz de confusión del modelo con un tamaño de imagen de 220x220 y 50 épocas con batch size de 8.

4.4.2 Modelo CNN-RNN para Reconocimiento de Actividades

En esta sección, se presenta el segundo modelo desarrollado para el reconocimiento de actividades en secuencias de vídeo. Este modelo combina una Red Neuronal Convolucional (CNN) con una Red Neuronal Recurrente (RNN), lo cual permite aprovechar las ventajas de ambas arquitecturas.

En el contexto del reconocimiento de actividades en vídeo, las Redes Neuronales Convolucionales (CNNs) se utilizan para extraer características espaciales de cada frame individual del vídeo. Las CNNs aplican filtros a través de sus capas de convolución para detectar características locales como bordes, texturas y patrones en diferentes regiones de una imagen. Estas características locales se combinan y se reducen a través de capas de pooling, las cuales disminuyen la dimensionalidad de las características y aumentan la eficiencia computacional al tiempo que mantienen la información relevante. Además, las capas densas, que son capas totalmente conectadas, se utilizan para procesar y combinar las características extraídas por las capas de convolución para realizar tareas de clasificación o detección.

Por otro lado, las Redes Neuronales Recurrentes (RNNs) están diseñadas para trabajar con datos secuenciales, como series temporales o secuencias de vídeo, donde la información presente en un paso de tiempo puede depender de la información en pasos anteriores. Las RNNs logran esto mediante la utilización de conexiones recurrentes que permiten mantener un estado oculto que se actualiza en cada paso de tiempo. En este modelo, se utiliza una variante específica de RNN llamada Gated Recurrent Unit (GRU). Las GRUs son conocidas por su capacidad para capturar dependencias temporales a largo plazo y por ser más eficientes computacionalmente que las tradicionales LSTM (Long Short-Term Memory).

La combinación de CNNs y RNNs en este modelo permite no solo analizar la información contenida en cada frame del vídeo de manera eficiente, sino también tener en cuenta la relación temporal entre los frames consecutivos. Esta integración es crucial para una correcta identificación de actividades en vídeo, ya que muchas actividades no pueden ser determinadas únicamente por un único frame, sino por la secuencia de movimientos y cambios en los frames a lo largo del tiempo. Esta dualidad en la estructura del modelo está diseñada para mejorar la precisión y la capacidad de generalización en el reconocimiento de actividades.

4.4.2.1 Arquitectura del Modelo

El modelo CNN-RNN para el reconocimiento de actividades en secuencias de vídeo se construye utilizando TensorFlow y Keras. La arquitectura del modelo combina capas convolucionales (CNN) para la extracción de características espaciales y una capa de Gated Recurrent Unit (GRU) para capturar dependencias temporales. A continuación, se describe en detalle la estructura del modelo:

- **Capas Convolucionales:**
  - **Primera capa convolucional:** Conv2D con 64 filtros, un tamaño de kernel de 3×3 y padding 'same', seguida de otra capa Conv2D con las mismas configuraciones.
  - **Batch Normalization:** Se aplica BatchNormalization después de las capas convolucionales para normalizar las activaciones, lo que acelera el entrenamiento y mejora la estabilidad del modelo.

  - **Max Pooling:** Una capa de MaxPooling2D reduce la dimensionalidad de las características extraídas, manteniendo la información más relevante y reduciendo la carga computacional.

  Este patrón se repite con capas convolucionales adicionales que aumentan progresivamente el número de filtros:

  - Segunda serie de capas convolucionales con 128 filtros.
  - Tercera serie de capas convolucionales con 256 filtros.
  - Cuarta serie de capas convolucionales con 512 filtros.

- **Capa de Flatten:** Después de las capas convolucionales, se incluye una capa Flatten que transforma la salida tridimensional en un vector unidimensional, preparando los datos para las capas densas y recurrentes.

- **Capa Recurrente (GRU):** Se utiliza una capa Gated Recurrent Unit (GRU) con 64 unidades para capturar las dependencias temporales en las secuencias de vídeo.

- **Capas Densas:** A continuación, se incluyen varias capas densas (totalmente conectadas) con la función de activación ReLU, que introducen no linealidad en el modelo y permiten la combinación de las características extraídas para realizar la clasificación. Se utiliza Dropout en estas capas para reducir el riesgo de sobreajuste.

  - **Primera capa densa:** 1024 unidades con activación ReLU y Dropout del 50%.
  - **Segunda capa densa:** 512 unidades con activación ReLU y Dropout del 50%.
  - **Tercera capa densa:** 128 unidades con activación ReLU y Dropout del 50%.
  - **Cuarta capa densa:** 64 unidades con activación ReLU y Dropout del 50%.

- **Capa de Salida:** La última capa es una capa densa con activación softmax, diseñada para la clasificación en el número de clases especificadas por `NUM_CLASSES`. La función softmax transforma las salidas de la red en un vector de probabilidades, donde cada valor es la probabilidad de pertenecer a una de las clases.

4.4.2.2 Esquema de la Arquitectura del Modelo

Para ilustrar visualmente la arquitectura del modelo descrito, a continuación se presenta un esquema básico:

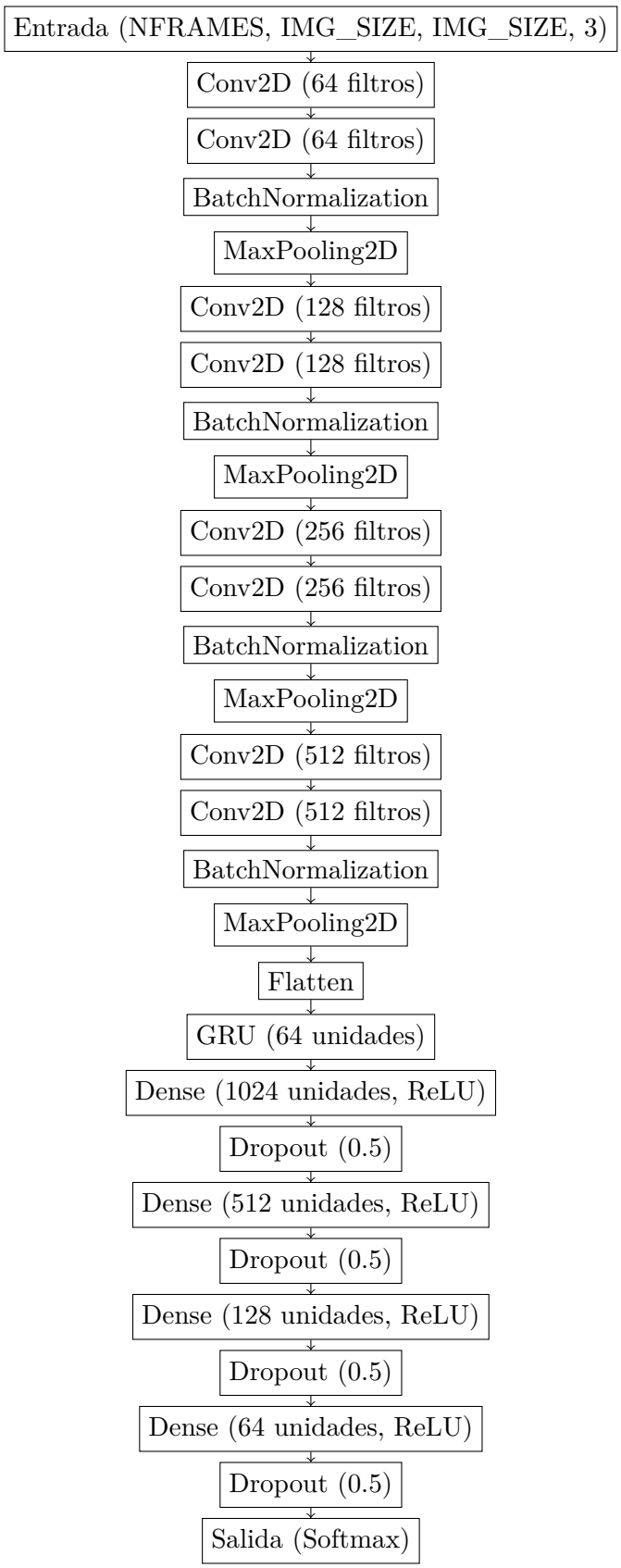

Figura 4.46: Esquema de la arquitectura del modelo CNN-RNN para reconocimiento de actividades.

4.4.2.3 Descripción Detallada de las Capas del Modelo

Cada capa en la arquitectura del modelo CNN-RNN desempeña una función esencial en el procesamiento y clasificación de secuencias de vídeo. A continuación, se explica el propósito y configuración de cada capa de manera detallada:

- **Capas Conv2D:** Estas capas aplican operaciones de convolución para extraer características espaciales de los frames de vídeo. Se utilizan filtros de tamaño 3×3 y padding 'same' para mantener las dimensiones de salida. Cada serie de capas convolucionales se sigue de una capa BatchNormalization y una capa MaxPooling2D.
- **Capa BatchNormalization:** Normaliza las activaciones de las capas convolucionales anteriores para estabilizar y acelerar el aprendizaje. Ajusta la media y la varianza de las activaciones.
- **Capa MaxPooling2D:** Reduce la dimensionalidad de las características extraídas, manteniendo la información más relevante y reduciendo la carga computacional. Ayuda a reducir el riesgo de sobreajuste.
- **Capa Flatten:** Transforma la salida tridimensional de las capas convolucionales en un vector unidimensional. Este proceso es crucial para preparar los datos para las capas densas y la capa recurrente.
- **Capa GRU:** La capa Gated Recurrent Unit (GRU) con 64 unidades captura las dependencias temporales en las secuencias de vídeo. Es más eficiente computacionalmente que las LSTM y puede manejar dependencias a largo plazo en los datos.
- **Capas Densas:**
  - **Primera capa densa:** 1024 unidades con activación ReLU. Incluye Dropout del 50% para reducir el sobreajuste.
  - **Segunda capa densa:** 512 unidades con activación ReLU. Incluye Dropout del 50%.
  - **Tercera capa densa:** 128 unidades con activación ReLU. Incluye Dropout del 50%.
  - **Cuarta capa densa:** 64 unidades con activación ReLU. Incluye Dropout del 50%.
- **Capa de Salida:** La última capa es una capa densa con activación softmax, diseñada para la clasificación en el número de clases especificadas por `NUM_CLASSES`. La función softmax transforma las salidas de la red en un vector de probabilidades, donde cada valor es la probabilidad de pertenecer a una de las clases.

4.4.2.4 Distribución de Datos para Entrenamiento, Validación y Pruebas

Antes de proceder al preprocesamiento y entrenamiento del segundo modelo, se ha realizado una distribución de los vídeos en conjuntos de entrenamiento, validación y pruebas. Para asegurar la validez del entrenamiento y la evaluación del modelo, es crucial mantener una representación equitativa de cada clase en los tres conjuntos. Se ha utilizado una proporción

de 60/20/20 para dividir los vídeos, lo que significa que aproximadamente el 60% de los vídeos de cada clase se asignan al entrenamiento, el 20% al conjunto de validación y el 20% al conjunto de pruebas. Este enfoque es consistente con el utilizado en el primer modelo, asegurando así una metodología uniforme a lo largo de ambos experimentos.

La siguiente tabla muestra la distribución exacta de vídeos por clase, incluyendo el total de vídeos, el número asignado al entrenamiento, validación y al conjunto de pruebas:

Clase	**Total**	**Entrenamiento**	**Validación**	**Tests**
Cleandishes	303	182	60	61
Cleanup	304	182	60	62
Cut	142	85	28	29
Stir	382	229	76	77
Usestove	77	46	15	16

Tabla 4.2: Distribución de vídeos por clase del segundo modelo para los conjuntos de entrenamiento, validación y pruebas, basada en una división estratificada de 60/20/20.

Esta estrategia de división asegura que cada conjunto refleje fielmente la distribución de las clases en el conjunto de datos completo, lo cual es vital para prevenir el sesgo en el entrenamiento y validar la capacidad del modelo para generalizar a nuevos datos. Durante el entrenamiento, las muestras se seleccionan y se presentan al modelo de manera aleatoria, garantizando así que cada época del entrenamiento exponga al modelo a diferentes combinaciones de datos. Esta aleatorización es crucial para prevenir el sobreajuste y asegurar que el modelo aprenda a generalizar a partir de los patrones en los datos en lugar de memorizar el orden específico de presentación.

4.4.2.5 Preprocesamiento de Datos y Entrenamiento

El preprocesamiento de los datos para el segundo modelo sigue un procedimiento similar al del primer modelo, detallado en la sección 4.4.1.5. Este procedimiento implica la lectura y preparación de los frames que componen cada vídeo, utilizando las funciones de OpenCV para el manejo de vídeos y TensorFlow para el ajuste de tamaño y padding con relleno de los frames.

Para facilitar la división de los vídeos en frames y su preprocesamiento, se utiliza la clase `FrameGenerator`, la cual asegura una carga eficiente de los frames en memoria según sean necesarios para el entrenamiento, validación y prueba del modelo.

El proceso de entrenamiento del modelo se realiza con un conjunto de datos configurado para el rendimiento mediante técnicas de caching, shuffling y prefetching, ya descritas en la sección 4.4.1.5. Se emplea un enfoque de entrenamiento por batch, con la configuración de hiperparámetros definidos al inicio del script.

El proceso de entrenamiento y validación ha sido monitoreado utilizando métricas de precisión y pérdida. Se han aplicado técnicas de regularización como Dropout para prevenir el sobreajuste, y se han utilizado callbacks de Keras para realizar ajustes dinámicos durante el entrenamiento.

4.4.2.6 Test 1: Evaluación con vídeos de 15 frames

Este test examina el comportamiento del modelo al procesar vídeos con una duración de 15 frames. El objetivo es evaluar la capacidad del modelo para manejar secuencias de vídeo cortas, centrando la atención en cómo se adapta a una cantidad limitada de información temporal por vídeo. Este tipo de evaluación es crítico para determinar la eficacia del modelo en escenarios donde solo se disponen de fragmentos breves de actividad, lo cual es común en aplicaciones de monitoreo en tiempo real.

4.4.2.6.1 Prueba 1: 50 épocas La primera prueba de este test se ha realizado con un total de 50 épocas. Los resultados, mostrados en las figuras 4.47 y 4.48, indican cómo el modelo se comporta a lo largo de las épocas de entrenamiento y validación.

La gráfica de precisión del modelo (Figura 4.47) muestra que la precisión de entrenamiento presenta una tendencia ascendente, alcanzando un máximo de aproximadamente 0.45, mientras que la precisión de validación también muestra mejoras, pero con fluctuaciones considerables que sugieren cierta variabilidad en la capacidad del modelo para generalizar a datos no vistos previamente. Esta observación es crítica, ya que destaca áreas potenciales para ajustar el modelo a fin de mejorar su estabilidad y rendimiento en la generalización.

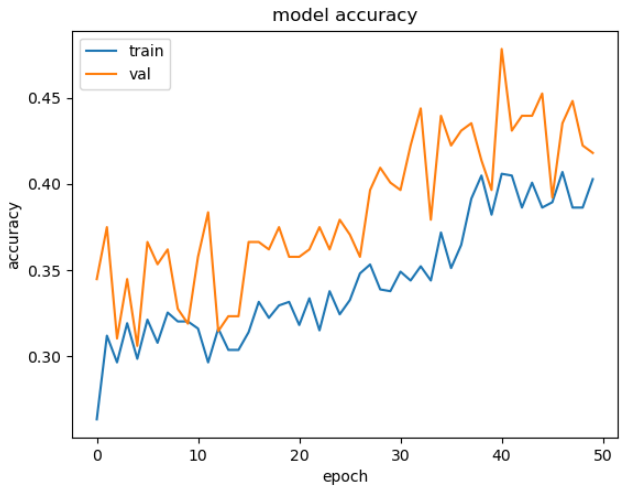

Figura 4.47: Precisión del modelo con vídeos de 15 frames y 50 épocas de entrenamiento.

En cuanto a la pérdida, la gráfica (Figura 4.48) revela una disminución estable en la pérdida de entrenamiento a lo largo de las épocas, indicando una buena convergencia del modelo. Sin embargo, la pérdida de validación, aunque disminuye, muestra picos ocasionales que pueden ser indicativos de sobreajuste o de necesidad de ajustes en los hiperparámetros.

La matriz de confusión obtenida (Figura 4.49) proporciona información detallada sobre la precisión de clasificación del modelo en las diferentes categorías de actividades. La clase Stir muestra la tasa más alta de clasificación correcta, aunque se observa una tendencia del modelo a confundir actividades de la clase Cleandishes y la clase Cleanup con la clase Stir. Este patrón de clasificación errónea como clase Stir indica que el modelo puede estar

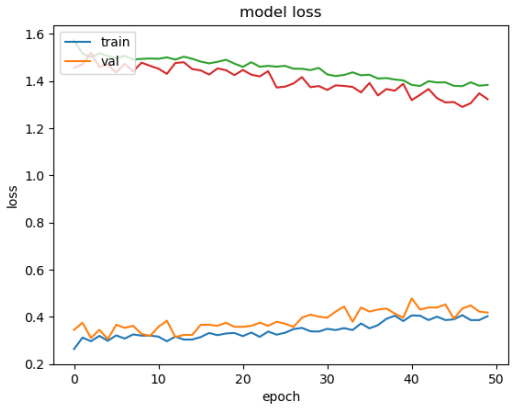

Figura 4.48: Pérdida del modelo con vídeos de 15 frames y 50 épocas de entrenamiento.

sobreajustando hacia características predominantes de esta clase, lo cual podría mitigarse ajustando el balance de clases en el conjunto de entrenamiento o refinando la extracción de características para mejorar la distinción entre actividades similares.

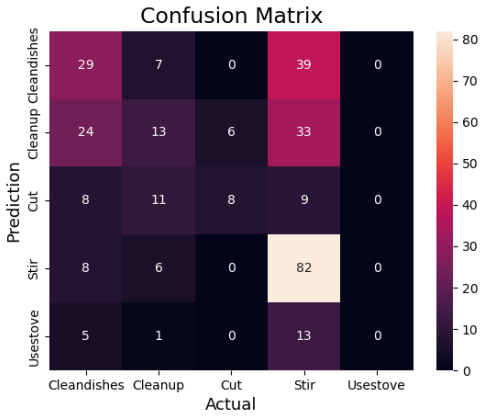

Figura 4.49: Matriz de confusión para la prueba de 50 épocas con vídeos de 15 frames.

Estos resultados demuestran que, aunque el modelo puede aprender de secuencias cortas de vídeo, la variabilidad en la precisión y pérdida de validación, junto con la tendencia a confundir ciertas clases con Stir, sugieren que aún hay espacio para mejorar la robustez y la capacidad de generalización del modelo.

4.4.2.6.2 Prueba 2: 100 épocas La segunda prueba del test con vídeos de 15 frames se ha extendido a 100 épocas, con el propósito de evaluar la capacidad de adaptación y mejora del modelo a lo largo de un periodo de entrenamiento más prolongado. Esta prueba ayuda a entender cómo el modelo maneja la generalización y la estabilidad de aprendizaje con un mayor número de iteraciones.

Los resultados mostrados en las figuras 4.50 y 4.51 subrayan un notable avance en el comportamiento del modelo. La precisión de entrenamiento (Figura 4.50) muestra una mejora continua, elevándose hasta aproximadamente un 0.65, lo que indica que el modelo se ha ajustado de manera efectiva a los datos de entrenamiento. La precisión de validación, por su parte, acompaña esta tendencia con un incremento consistente y alcanza un nivel cercano al de entrenamiento hacia el final de las épocas, sugiriendo una mejora en la capacidad de generalización del modelo.

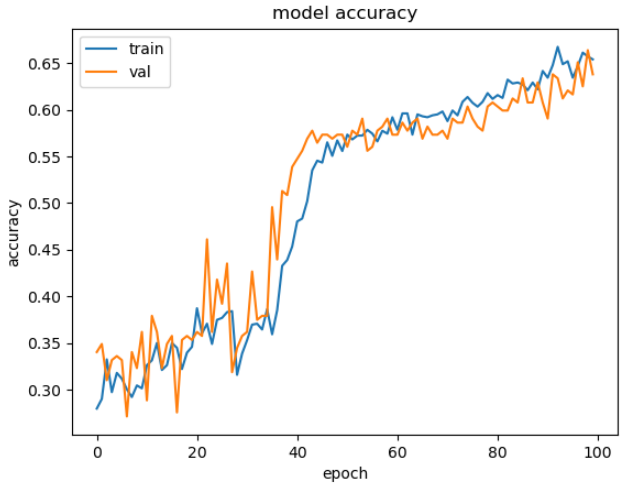

Figura 4.50: Precisión del modelo con vídeos de 15 frames y 100 épocas de entrenamiento.

En relación a la pérdida, la gráfica (Figura 4.51) indica que la pérdida de entrenamiento ha disminuido de forma constante, manteniendo una línea relativamente plana y baja hacia las últimas épocas, lo que refleja una convergencia eficiente del modelo. La pérdida de validación ha seguido una trayectoria descendente similar, aunque con algunos picos que sugieren momentos de inestabilidad en el aprendizaje. Estos picos podrían implicar la necesidad de ajustar aún más los hiperparámetros para lograr una estabilidad más robusta en fases futuras de entrenamiento.

La matriz de confusión obtenida para esta prueba (Figura 4.52) muestra una distribución más equilibrada en la clasificación de actividades que en la prueba anterior en la que se usaban 50 épocas, aunque aún con ciertas tendencias a confundir las clases Cleandishes y Cleanup con otras actividades. Especialmente notable es la mejora en la correcta identificación de la clase Stir y la reducción en la confusión con otras categorías, lo que indica una mejora en la discriminación del modelo entre actividades visuales similares.

Estos resultados consolidan la capacidad del modelo para aprender efectivamente de secuencias cortas de vídeo a lo largo de un extenso periodo de entrenamiento, mostrando una mejora continua tanto en precisión como en pérdida, lo cual es prometedor para aplicaciones prácticas donde la consistencia y la confiabilidad son cruciales.

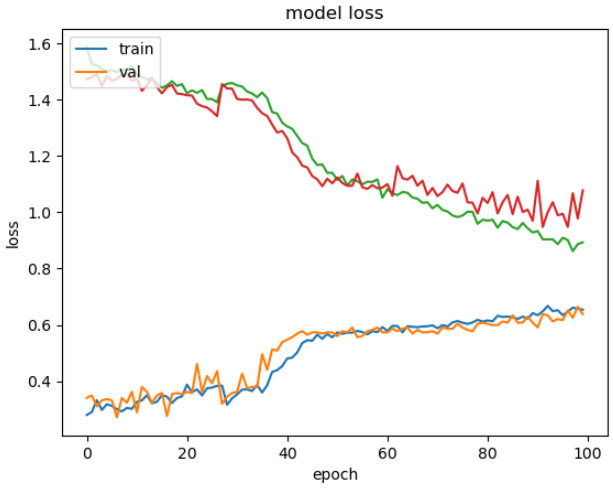

Figura 4.51: Pérdida del modelo con vídeos de 15 frames y 100 épocas de entrenamiento.

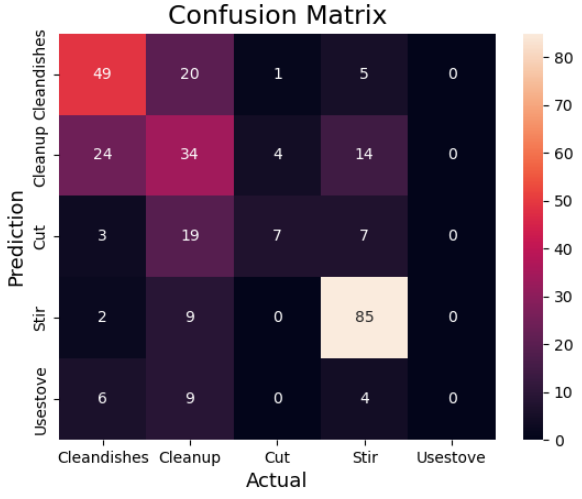

Figura 4.52: Matriz de confusión para la prueba de 100 épocas con vídeos de 15 frames.

4.4.2.6.3 Prueba 3: 200 épocas La tercera prueba de este test ha extendido el entrenamiento del modelo a 200 épocas, enfocándose en observar la durabilidad y estabilidad del aprendizaje a largo plazo con vídeos de 15 frames. Esta evaluación extensa es vital para determinar la robustez del modelo bajo condiciones de entrenamiento prolongado y para verificar la consistencia de su rendimiento en generalización.

La precisión de entrenamiento, como se muestra en la figura 4.53, ha continuado mejorando ligeramente hasta estabilizarse alrededor de un 0.70, reflejando un ajuste sólido y consistente del modelo a los datos de entrenamiento. La precisión de validación también se ha estabilizado, siguiendo de cerca a la de entrenamiento, lo que indica que el modelo ha alcanzado un nivel de generalización muy competente. Esta convergencia entre las curvas de precisión de entrenamiento y validación sugiere que el modelo ha madurado en términos de aprender y adaptarse a las variaciones dentro de los datos sin sobreajustarse significativamente.

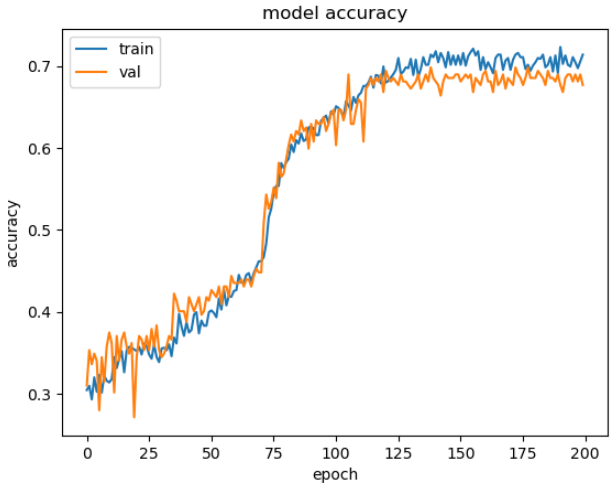

Figura 4.53: Precisión del modelo con vídeos de 15 frames y 200 épocas de entrenamiento.

La gráfica de pérdida (Figura 4.54) ilustra una disminución progresiva en la pérdida de entrenamiento que llega a estabilizarse en un nivel bajo, indicando una buena convergencia y eficiencia en el aprendizaje del modelo. Por otro lado, la pérdida de validación muestra una estabilidad a partir de la mitad del entrenamiento, con fluctuaciones menores que reflejan la capacidad del modelo para manejar nuevos datos de manera efectiva. Estos resultados son un testimonio de la capacidad del modelo para sostener un rendimiento sólido y confiable a lo largo de un periodo extendido.

La matriz de confusión resultante de esta prueba de 200 épocas revela patrones de clasificación que subrayan tanto las fortalezas como las áreas de mejora para el modelo (Figura 4.55). De todas las pruebas realizadas, esta ha mostrado los mejores resultados en términos de precisión y capacidad de generalización, particularmente para la categoría Stir, que se clasifica con alta precisión. Este éxito destaca la eficacia del modelo en reconocer y generalizar bien esta actividad, lo cual es un logro significativo dado el número limitado de frames por vídeo.

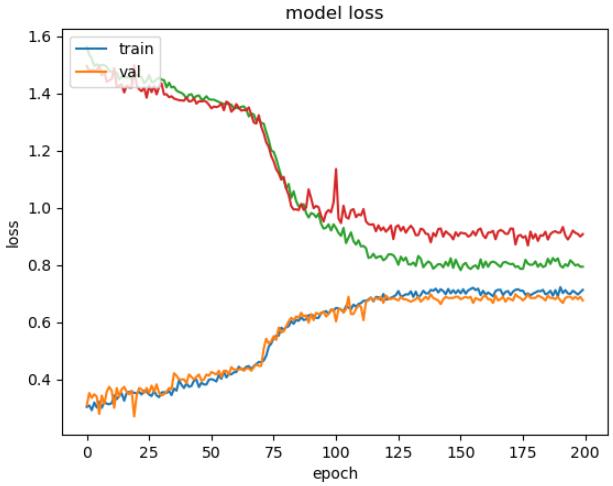

Figura 4.54: Pérdida del modelo con vídeos de 15 frames y 200 épocas de entrenamiento.

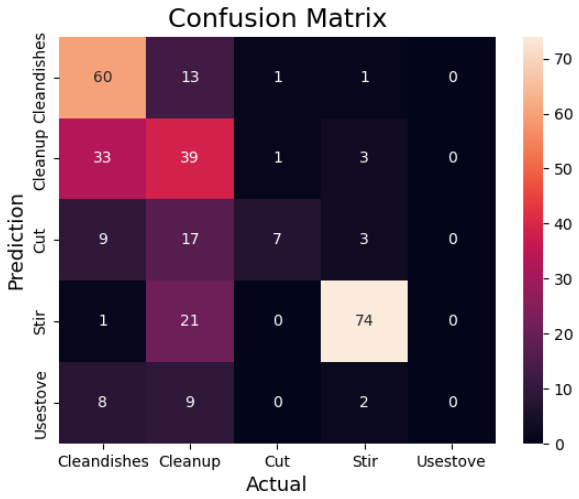

Figura 4.55: Matriz de confusión para la prueba de 200 épocas con vídeos de 15 frames.

Sin embargo, las confusas clasificaciones entre las categorías Cleandishes y Cleanup reflejan una debilidad en la diferenciación de actividades que comparten características visuales o temporales similares. Es evidente la subóptima precisión en estas categorías y puede llevar a errores significativos en aplicaciones prácticas, especialmente en entornos de monitoreo en tiempo real donde cada actividad tiene implicaciones distintas.

Para abordar estas áreas de mejora, se podría considerar la integración de técnicas como el aumento de datos dirigido específicamente a estas categorías, lo cual ayudaría a aumentar la diversidad de los ejemplos de entrenamiento y a fortalecer la robustez del modelo frente a variaciones en los datos.

La consistencia en las métricas de precisión y pérdida, combinada con los detalles de la

matriz de confusión de esta prueba de 200 épocas, muestra la capacidad del modelo para funcionar eficazmente en contextos dinámicos y variados. Especialmente destacable es la alta precisión en la categoría Stir, lo que refleja la madurez del modelo en generalizar adecuadamente los datos de entrenamiento a situaciones reales, crucial para aplicaciones en tiempo real donde se valora la precisión y la fiabilidad. No obstante, las confusiones recurrentes entre las categorías Cleandishes y Cleanup indican la necesidad de mejorar la discriminación entre actividades similares. Estos hallazgos sugieren áreas claras para futuras mejoras y resaltan la importancia de ajustar la arquitectura del modelo y los procesos de preprocesamiento para optimizar el rendimiento general en implementaciones prácticas.

4.4.2.7 Test 2: Evaluación con vídeos de 20 frames

Este test se centra en el análisis del comportamiento del modelo al procesar vídeos de 20 frames de duración. Al aumentar el número de frames respecto al primer test, se pretende examinar cómo maneja el modelo un volumen de información temporal ligeramente superior. Este incremento en el número de frames permite explorar la capacidad del modelo para capturar y procesar patrones más complejos y sutiles en las secuencias de vídeo, lo que es crucial para aplicaciones que requieren un análisis más detallado y profundo de las actividades registradas.

4.4.2.7.1 Prueba 1: 50 épocas La primera prueba del Test 2 se ha realizado durante 50 épocas. Este período de tiempo ha sido seleccionado para proporcionar una comparación directa con los tests anteriores, permitiendo evaluar cómo se adapta el modelo a un incremento en la duración de los vídeos.

La gráfica de precisión del modelo (Figura 4.56) indica un comportamiento fluctuante, con la precisión de entrenamiento alcanzando un pico de aproximadamente 0.36, mientras que la precisión de validación oscila en un rango más bajo. Este patrón sugiere que el modelo lucha por generalizar efectivamente a partir de los datos no vistos, lo cual podría deberse a la complejidad añadida de los vídeos más largos.

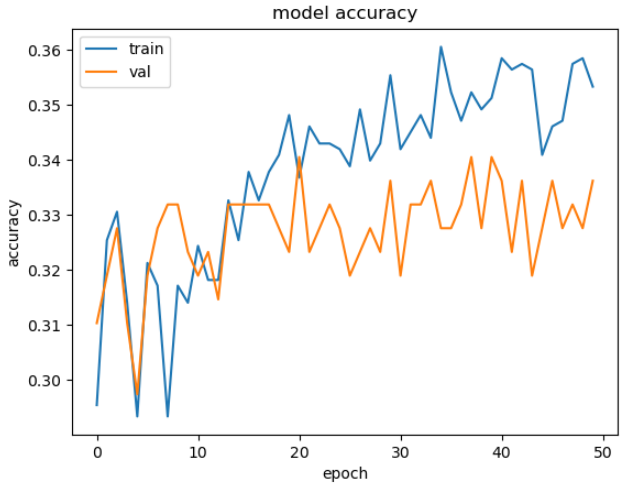

Figura 4.56: Precisión del modelo con vídeos de 20 frames y 50 épocas de entrenamiento.

En cuanto a la pérdida, los resultados (Figura 4.57) muestran una disminución constante en la pérdida de entrenamiento, reflejando una buena convergencia del modelo en este aspecto. Sin embargo, la pérdida de validación presenta fluctuaciones significativas, lo que indica posibles problemas de sobreajuste o dificultades del modelo para ajustarse a la variabilidad incrementada en los datos de validación.

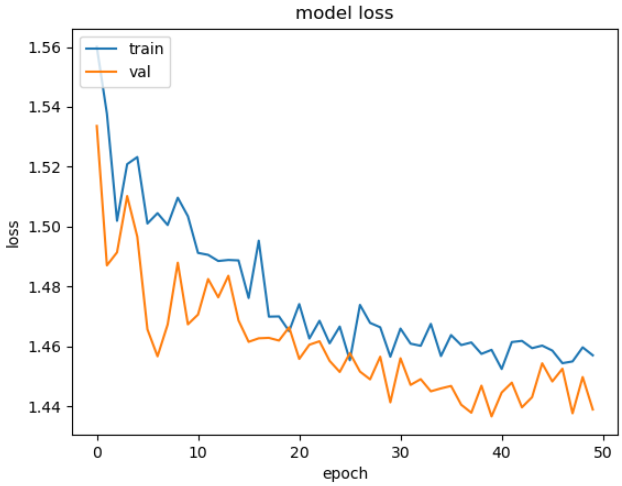

Figura 4.57: Pérdida del modelo con vídeos de 20 frames y 50 épocas de entrenamiento.

La matriz de confusión obtenida de esta prueba (Figura 4.58) revela una distribución de clasificaciones que indica un desafío en la capacidad del modelo para diferenciar eficazmente entre ciertas actividades, con notables confusiones entre las categorías Cleandishes y Cleanup, y una clasificación más precisa en la categoría Stir. Esta información es crucial para

identificar dónde el modelo necesita refinamientos adicionales para mejorar su discriminación entre actividades visualmente o temporalmente similares.

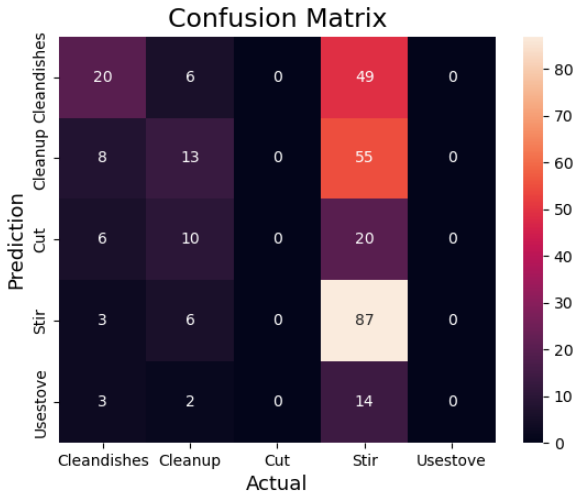

Figura 4.58: Matriz de confusión para la prueba de 50 épocas con vídeos de 20 frames.

Estos resultados sugieren que, si bien el modelo puede aprender patrones a partir de secuencias más largas, aún existen desafíos significativos en términos de generalización y estabilidad que deben abordarse para mejorar su rendimiento en condiciones reales. Sería interesante la exploración de ajustes en la arquitectura del modelo o en los hiperparámetros para mejorar la capacidad de generalización con secuencias de vídeo más extensas.

4.4.2.7.2 Prueba 2: 200 épocas En la segunda prueba del Test 2 se ha extendido el período de entrenamiento del modelo a 200 épocas, con el objetivo de profundizar en la evaluación de la capacidad de aprendizaje y generalización del modelo con vídeos de 20 frames. Este período más extenso permite observar las dinámicas de aprendizaje a largo plazo, particularmente cómo el modelo se adapta y estabiliza frente a un volumen de datos incrementado y posiblemente más complejo.

La precisión de entrenamiento (Figura 4.59) muestra un ascenso notable, estabilizándose en torno al 0.65, lo que refleja una capacidad robusta del modelo para aprender de los datos de entrenamiento. Por otro lado, la precisión de validación muestra un comportamiento más constante en comparación con la primera prueba, oscilando alrededor de 0.55, lo cual es indicativo de una mejora en la generalización respecto a pruebas más cortas. Este incremento en la precisión de validación a lo largo de las 200 épocas sugiere que el modelo ha comenzado a adaptarse de manera más efectiva a las variabilidades inherentes a los datos no vistos.

En relación con la pérdida, la gráfica (Figura 4.60) indica una reducción significativa y consistente en la pérdida de entrenamiento, manteniéndose en niveles bajos hacia el final del período de entrenamiento. Esto demuestra una excelente convergencia y eficiencia del modelo en el procesamiento de los datos de entrenamiento. La pérdida de validación, por su parte, muestra una estabilidad notable a lo largo del tiempo, sin los picos agudos observados en períodos de entrenamiento más cortos, lo cual es un buen indicador de que el modelo está

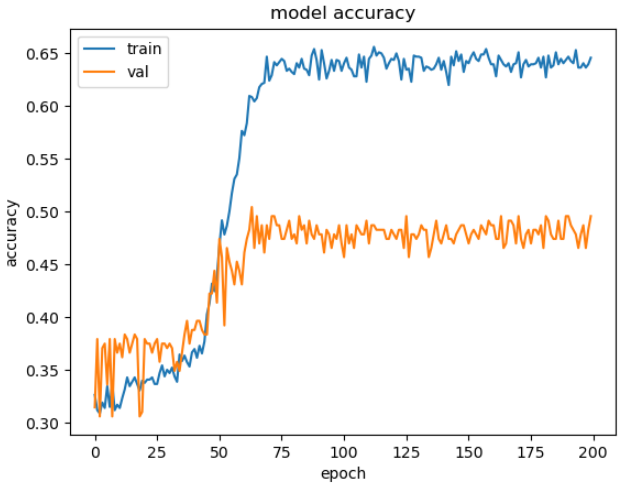

Figura 4.59: Precisión del modelo con vídeos de 20 frames y 200 épocas de entrenamiento.

manejando bien el sobreajuste y adaptándose de manera adecuada a nuevas situaciones.

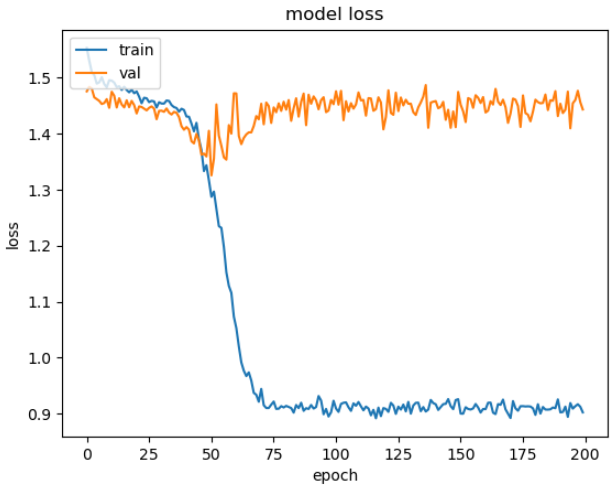

Figura 4.60: Pérdida del modelo con vídeos de 20 frames y 200 épocas de entrenamiento.

Al observar la matriz de confusión para esta prueba (Figura 4.61), se aprecia que el modelo todavía enfrenta desafíos en la discriminación precisa entre algunas categorías. Aunque hay una mejora en la clasificación de la categoría Stir, las confusiones entre las categorías Cleandishes y Cleanup persisten, lo que sugiere que el modelo todavía lucha con actividades que comparten rasgos visuales similares. Estas dificultades son consistentes con las observaciones de pruebas anteriores, lo que indica que mientras hay avances en algunas áreas, las confusiones entre ciertas categorías siguen siendo un área importante para futuras mejoras.

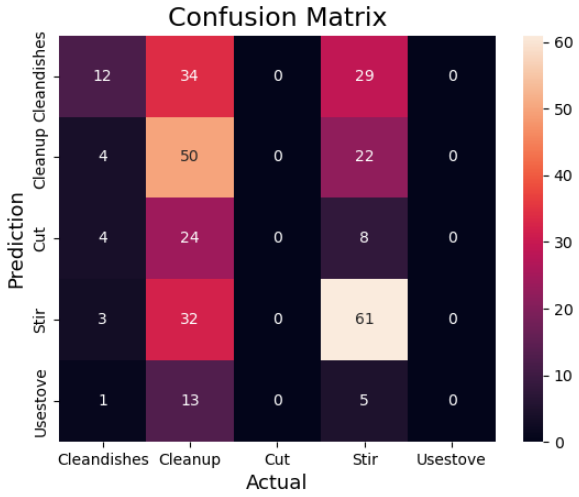

Figura 4.61: Matriz de confusión para la prueba de 200 épocas con vídeos de 20 frames.

Estos resultados indican que, aunque hay una evolución positiva en la precisión y en la gestión de la pérdida, el modelo aún requiere ajustes para mejorar su capacidad de diferenciación entre actividades visualmente similares. Este análisis subraya la importancia de continuar refinando tanto la arquitectura del modelo como las técnicas de procesamiento y entrenamiento para optimizar la generalización y la precisión en aplicaciones prácticas, especialmente aquellas que operan en tiempo real donde la precisión y la confiabilidad son esenciales.

5 Discusión

5.1 Interpretación de Resultados

En esta sección se analizan los resultados obtenidos durante el entrenamiento y la validación de los modelos propuestos, en el contexto de los objetivos planteados y la literatura existente. Se presentan las observaciones relevantes sobre el rendimiento de los modelos, así como comparaciones con benchmarks o modelos similares disponibles en la literatura. Los resultados se evaluarán en función de la precisión alcanzada, destacando los puntos fuertes y las limitaciones, y proponiendo posibles mejoras para futuras investigaciones.

5.1.1 Resumen de Precisión de los Modelos

A continuación, se presentan las tablas que resumen las precisiones obtenidas en las diferentes pruebas para los modelos ConvLSTM y CNN-RNN.

5.1.1.1 Modelo ConvLSTM

El modelo ConvLSTM muestra variabilidad en la precisión dependiendo de las configuraciones de tamaño de imagen y número de épocas. La tabla 5.1 resume las precisiones obtenidas en las diferentes pruebas:

Test	**Configuración**	**Precisión (%)**
Test 1: EarlyStopping	Prueba 1: patience=0	16.67
	Prueba 2: patience=0.002	23.33
	Prueba 3: Sin EarlyStopping	23.33
Test 2: Modificación de Hiperparámetros y Arquitectura	-	32
Test 3	Prueba 1: Batch Size de 4	50
	Prueba 2: Batch Size de 8	67
	Prueba 3: Batch Size de 16	32
Test 4: Batch Size de 16 y 50 Épocas	-	46
Test 5: Imagen 200x200	Prueba 1: 25 Épocas, Batch Size de 8	32
	Prueba 2: 50 Épocas, Batch Size de 4	32
Test 6: Imagen 220x220	Prueba 1: 25 Épocas, Batch Size de 8	32
	Prueba 2: 50 Épocas, Batch Size de 8	32

Tabla 5.1: Precisión obtenida en las diferentes pruebas del modelo ConvLSTM.

Analizando los resultados obtenidos por el modelo ConvLSTM, se observa que la precisión varía considerablemente según los hiperparámetros configurados. En el Test 1, la implementación de la técnica de EarlyStopping con diferentes valores de *patience* muestra que un *patience* más alto mejora ligeramente la precisión en test. La modificación de hiperparámetros en el Test 2 ha resultado en una mejora significativa de la precisión, alcanzando un 32%. En el Test 3, la variación del tamaño de batch muestra que un tamaño de batch de 8 proporciona la mayor precisión (67%). Sin embargo, aumentar o disminuir significativamente el tamaño del batch ha afectado negativamente la precisión. El Test 4, con un batch size de 16 y 50 épocas, se ha obtenido una precisión de 46%, lo que sugiere que aumentar el número de épocas puede ayudar a mejorar la precisión. Los Test 5 y 6 han demostrado que cambiar el tamaño de la imagen no afecta significativamente la precisión, que se mantiene constante en 32%.

5.1.1.2 Modelo CNN-RNN

El modelo CNN-RNN también ha presentado variaciones en su rendimiento. La tabla 5.2 resume las precisiones obtenidas en las diferentes pruebas:

Test	**Configuración**	**Precisión (%)**
Test 1: 15 frames	Prueba 1: 50 épocas	44
	Prueba 2: 100 épocas	58
	Prueba 3: 200 épocas	60
Test 2: 20 frames	Prueba 1: 50 épocas	40
	Prueba 2: 200 épocas	41

Tabla 5.2: Precisión obtenida en las diferentes pruebas del modelo CNN-RNN.

En el análisis de los resultados del modelo CNN-RNN, se destaca que la precisión mejora con el incremento del número de épocas, especialmente con secuencias de 15 frames. La precisión ha aumentado de 44% a 60% al pasar de 50 a 200 épocas en el Test 1. Sin embargo, con vídeos de 20 frames, la mejora en precisión es menos significativa, pasando de 40% a 41% al extender las épocas de 50 a 200. Esto sugiere que el modelo tiene un límite en su capacidad de mejora con secuencias más largas, posiblemente debido a la mayor complejidad temporal.

5.1.2 Comparación de Modelos

Al comparar los dos modelos, el modelo CNN-RNN ha mostrado un mejor rendimiento general, alcanzando la mayor precisión de 60% con la configuración de 15 frames y 200 épocas, en comparación con la máxima precisión de 67% del modelo ConvLSTM con una configuración de batch size de 8 en 25 épocas. Aunque el modelo ConvLSTM ha alcanzado una precisión más alta en una configuración específica, el modelo CNN-RNN muestra una consistencia mejorada y un rendimiento robusto con configuraciones variadas.

El análisis detallado de los resultados de ambos modelos revela varios aspectos importantes:

5.1.2.0.1 Consistencia de Resultados: El modelo CNN-RNN evidencia un rendimiento más consistente a lo largo de diferentes configuraciones en comparación con el modelo ConvLSTM.

Esto es crucial para aplicaciones prácticas donde se necesita un rendimiento fiable independientemente de las variaciones en los datos de entrada.

5.1.2.0.2 Efecto del Número de Épocas: Se observa que aumentar el número de épocas generalmente mejora la precisión, especialmente en el modelo CNN-RNN. Sin embargo, en el modelo ConvLSTM, un aumento excesivo del tamaño de batch no siempre conduce a mejores resultados, indicando una necesidad de equilibrar los hiperparámetros cuidadosamente.

5.1.2.0.3 Tamaño de Imagen: Los resultados sugieren que cambiar el tamaño de la imagen no ha tenido un impacto significativo en la precisión del modelo ConvLSTM, mientras que el modelo CNN-RNN no ha sido evaluado con diferentes tamaños de imagen, lo cual puede ser una área de exploración futura.

5.1.2.0.4 Capacidad de Generalización: La capacidad del modelo CNN-RNN para mantener una precisión relativamente alta con diferentes configuraciones de número de frames y épocas sugiere una mejor capacidad de generalización en comparación con el modelo ConvLSTM.

5.1.3 Validez de los Modelos

En términos generales, se considera que un modelo de reconocimiento de actividades en vídeo es adecuado si alcanza una precisión superior al 70%. Ninguno de los modelos evaluados realmente ha alcanzado este umbral, lo que indica que hay margen para mejoras significativas. El modelo ConvLSTM ha logrado una precisión máxima del 67%, mientras que el modelo CNN-RNN ha logrado un 60%. Aunque estos resultados son verdaderamente prometedores, sugieren que ambos modelos necesitan ajustes adicionales y optimización para ser considerados altamente precisos y fiables en aplicaciones prácticas.

Estos resultados resaltan la importancia de la selección adecuada de hiperparámetros y la arquitectura del modelo en el rendimiento del reconocimiento de actividades en vídeo. Además, sugieren que, aunque los modelos basados en ConvLSTM pueden alcanzar precisiones altas en configuraciones específicas, los modelos CNN-RNN pueden ofrecer un rendimiento más robusto y consistente en diferentes escenarios.

5.2 Limitaciones y Retos

A pesar de los avances logrados, este trabajo enfrenta varias limitaciones y retos que han influido en los resultados obtenidos. Entre las principales limitaciones se encuentran:

5.2.1 Tamaño y Calidad del Dataset

El tamaño del dataset y la calidad de las anotaciones son factores cruciales que afectan el rendimiento del modelo. En este estudio, la disponibilidad limitada de datos etiquetados ha sido un desafío significativo. Idealmente, un dataset debería contener miles de ejemplos por clase para asegurar una correcta generalización del modelo. Sin embargo, el dataset utilizado en este estudio cuenta con un número considerablemente menor de ejemplos por clase, lo que puede limitar la capacidad del modelo para aprender patrones robustos y generalizables.

Además, la variabilidad en la calidad de las anotaciones puede introducir ruido en el proceso de entrenamiento, afectando la capacidad del modelo para generalizar adecuadamente. La construcción de datasets más grandes y consistentemente anotados sería esencial para mejorar el rendimiento del modelo en futuras investigaciones.

5.2.2 Características Visuales Similares

Los modelos implementados muestran dificultades para diferenciar entre actividades que comparten características visuales similares, como se observa en las matrices de confusión presentadas. Este problema puede atribuirse a la naturaleza intrínseca de las actividades, que a menudo involucran movimientos y objetos similares, dificultando la tarea de clasificación precisa. Una posible solución a este problema sería el uso de imágenes de mayor calidad y resolución, lo cual permitiría captar detalles más finos en los movimientos y posturas de, en este caso, las personas. Además, la integración de características adicionales, como la información temporal más detallada y el uso de modelos más complejos que puedan captar mejor estas sutiles diferencias, también podría ser beneficioso.

5.2.3 Capacidad de Generalización

Aunque se han aplicado técnicas de regularización como Dropout para prevenir el sobreajuste, los resultados de validación indican que los modelos aún pueden mejorar su capacidad de generalización. La implementación de técnicas más avanzadas de aumento de datos y el uso de modelos más complejos podrían ayudar a abordar este problema en futuros trabajos

5.2.4 Recursos Computacionales

El entrenamiento de modelos de deep learning requiere una cantidad significativa de recursos computacionales. Las limitaciones en el acceso a hardware de alto rendimiento, como GPUs o TPUs, pueden restringir la capacidad para realizar entrenamientos más extensos y optimizaciones de hiperparámetros, lo cual podría mejorar el rendimiento de los modelos. No obstante, hay varias áreas en las que se podrían realizar mejoras significativas con acceso a recursos computacionales avanzados:

5.2.4.0.1 Optimización del Tiempo de Entrenamiento Con hardware más potente, los tiempos de entrenamiento se pueden reducir considerablemente. Esto permitiría realizar múltiples iteraciones de prueba y error de manera más eficiente, ajustando los hiperparámetros con mayor precisión y mejorando así el rendimiento del modelo. Además, una reducción en los tiempos de entrenamiento facilita la implementación de técnicas avanzadas como la búsqueda de hiperparámetros automatizada (AutoML).

5.2.4.0.2 Aumento del Tamaño del Modelo Con acceso a recursos computacionales avanzados, sería posible entrenar modelos más grandes y complejos que pueden ofrecer mejores resultados. Las arquitecturas más profundas y sofisticadas, que actualmente no son viables debido a limitaciones de memoria y procesamiento, podrían ser implementadas, explorando así su potencial para mejorar la precisión del reconocimiento de actividades.

5.2.4.0.3 Procesamiento de Imágenes de Alta Resolución La capacidad de procesar imágenes de alta resolución, que podría mejorar significativamente la precisión del modelo al captar detalles más finos en los movimientos y posturas humanas, se vería aumentada. Con hardware avanzado, no solo sería factible manejar la mayor carga computacional, sino que también se podría explorar el impacto de diferentes resoluciones de imagen en el rendimiento del modelo.

En resumen, aunque este estudio ha logrado avances significativos en el reconocimiento de acciones mediante deep learning, la implementación de hardware avanzado y más potente, así como la utilización de servicios de computación en la nube, podría mitigar las actuales limitaciones y permitir un desarrollo más robusto y eficiente de los modelos en futuros trabajos. Estas mejoras no solo optimizarían los tiempos de entrenamiento y permitirían la exploración de modelos más complejos, sino que también abrirían nuevas posibilidades para el procesamiento de imágenes de alta resolución y la implementación de técnicas avanzadas de aumento de datos, mejorando así la precisión y la capacidad de generalización de los modelos en aplicaciones prácticas.

6 Conclusiones

En este trabajo de fin de grado, se han explorado diversas arquitecturas de deep learning para el reconocimiento de actividades humanas en secuencias de vídeo. Los modelos implementados, ConvLSTM y CNN-RNN, han sido evaluados exhaustivamente en términos de precisión, capacidad de generalización y consistencia de resultados bajo diferentes configuraciones de hiperparámetros y tamaños de datos. A continuación, se presentan las conclusiones más relevantes obtenidas a partir de este estudio.

6.1 Síntesis de Resultados

El modelo ConvLSTM, que combina convoluciones espaciales con memoria a largo plazo para capturar tanto características espaciales como temporales, ha mostrado una variabilidad significativa en sus resultados. La precisión máxima obtenida ha sido del 67% con una configuración específica de tamaño de batch y número de épocas. Sin embargo, este modelo presenta dificultades para mantener una precisión consistente a través de diferentes configuraciones, evidenciando una sensibilidad considerable a los hiperparámetros.

Por otro lado, el modelo CNN-RNN, que integra convoluciones para la extracción de características espaciales con unidades recurrentes para capturar dependencias temporales, manifiesta un rendimiento más robusto y consistente. La precisión máxima alcanzada es del 60% con una configuración de 15 frames y 200 épocas. Este modelo evidencia una mejor capacidad de generalización y una menor variabilidad en los resultados, lo que lo hace más adecuado para aplicaciones prácticas donde se requiere un rendimiento fiable.

6.2 Impacto de los Hiperparámetros y Configuraciones

El estudio ha resaltado la importancia crítica de la selección de hiperparámetros y configuraciones en el rendimiento de los modelos de deep learning. La variabilidad observada en la precisión del modelo ConvLSTM bajo diferentes configuraciones de tamaño de batch y número de épocas subraya la necesidad de una sintonización cuidadosa de estos parámetros. Además, se ha evidenciado que aumentar el número de épocas generalmente mejora la precisión, aunque esto debe equilibrarse con el riesgo de sobreajuste.

Para el modelo CNN-RNN, la consistencia de los resultados sugiere que esta arquitectura es más robusta frente a cambios en los hiperparámetros. No obstante, la evaluación no exhaustiva con diferentes tamaños de imagen para este modelo indica que futuras investigaciones podrían explorar este aspecto para optimizar aún más su rendimiento.

6.3 Limitaciones del Estudio

A pesar de los avances logrados, el estudio presenta varias limitaciones que han influido en los resultados obtenidos. La disponibilidad limitada de datos etiquetados y la variabilidad en la calidad de las anotaciones han sido desafíos significativos que pueden haber introducido ruido en el proceso de entrenamiento, afectando la capacidad de generalización de los modelos. Además, las restricciones en el acceso a hardware de alto rendimiento han limitado la capacidad para realizar entrenamientos más extensos y optimizaciones de hiperparámetros.

6.4 Implicaciones Prácticas

Los resultados de este estudio tienen importantes implicaciones prácticas para el desarrollo de sistemas de reconocimiento de acciones mediante deep learning. La robustez y consistencia demostradas por el modelo CNN-RNN lo hacen adecuado para aplicaciones en tiempo real, como la monitorización de seguridad y las interfaces de usuario interactivas. La capacidad de este modelo para mantener una precisión relativamente alta bajo diferentes configuraciones sugiere que es una opción viable para entornos dinámicos y variados.

6.5 Conclusión Final

En conclusión, este trabajo ha demostrado que, aunque ambos modelos tienen sus propias ventajas y limitaciones, el modelo CNN-RNN ofrece un rendimiento más robusto y consistente para el reconocimiento de actividades humanas en secuencias de vídeo. Las mejoras en la selección de hiperparámetros, el acceso a recursos computacionales avanzados y la integración de señales multimodales tienen el potencial de llevar estos modelos a niveles de precisión y aplicabilidad aún mayores, abriendo nuevas posibilidades para aplicaciones prácticas en diversas áreas.

7 Trabajos Futuros

7.1 Propuestas de Investigación Futura

El campo del reconocimiento de acciones humanas en imágenes RGB, utilizando técnicas avanzadas de deep learning, continúa ofreciendo numerosas oportunidades para investigación. A la luz de los resultados obtenidos en este trabajo, se identifican varias direcciones prometedoras para la investigación futura:

7.1.1 Exploración de Arquitecturas Más Profundas y Complejas

Una línea prometedora de investigación es la exploración de arquitecturas neuronales más profundas y complejas, como redes neuronales de transformers, que han mostrado resultados sobresalientes en otras áreas del deep learning. Estas arquitecturas pueden mejorar la capacidad de generalización del modelo, permitiendo que se desempeñe bien en situaciones inéditas y variaciones en los datos.

7.1.2 Métodos de Aprendizaje Semi-Supervisado y No Supervisado

La adopción de métodos de aprendizaje semi-supervisado y no supervisado podría superar las restricciones impuestas por la necesidad de grandes volúmenes de datos etiquetados. El aprendizaje semi-supervisado utiliza una combinación de un pequeño conjunto de datos etiquetados junto con un gran volumen de datos no etiquetados, mejorando el aprendizaje del modelo. Esta técnica es especialmente útil en escenarios donde la recopilación y etiquetado de datos es costosa o impracticable.

Por otro lado, el aprendizaje no supervisado, que no requiere datos etiquetados, emplea técnicas como el agrupamiento y la detección de anomalías para entender la estructura y distribución de los datos. Implementar estas metodologías podría revelar patrones ocultos en los datos visuales, mejorando la identificación de acciones complejas sin supervisión explícita.

7.1.3 Desarrollo de Modelos en Tiempo Real

El desarrollo de modelos capaces de operar en tiempo real con alta precisión es esencial para aplicaciones en áreas como la monitorización de seguridad y las interfaces de usuario interactivas. La capacidad de procesar y analizar vídeo en tiempo real permite intervenciones inmediatas en situaciones críticas, como la identificación de comportamientos sospechosos en áreas públicas o la interacción fluida en sistemas de realidad aumentada.

7.1.4 Integración de Señales Multimodales

La integración de señales multimodales, como audio, datos cinemáticos y las propias imágenes RGB, puede proporcionar una comprensión más rica y matizada de las situaciones. Por

ejemplo, en un entorno de seguridad, la combinación de audio con el análisis visual puede ayudar a identificar situaciones de emergencia, como alguien pidiendo ayuda. Del mismo modo, en interfaces de usuario interactivas, la sincronización de entrada gestual con comandos de voz puede permitir experiencias de usuario más naturales y efectivas. La investigación futura podría explorar cómo estas tecnologías multimodales se complementan para mejorar la precisión y robustez del reconocimiento de acciones complejas.

7.2 Mejoras en Metodología

La metodología utilizada en este TFG ha demostrado su eficacia; sin embargo, hay varias áreas susceptibles de mejora para aumentar la robustez y la efectividad del modelo propuesto:

7.2.1 Aumento de Datos

Una de las primeras sugerencias es la implementación de técnicas más avanzadas de aumento de datos, tales como transformaciones geométricas, variaciones de iluminación y técnicas de síntesis de datos. Estas técnicas pueden potenciar la habilidad del modelo para generalizar a partir de nuevas imágenes no vistas previamente y mitigar el sobreajuste.

7.2.2 Diversidad del Dataset

Incrementar la diversidad del dataset de entrenamiento para abarcar un rango más amplio de acciones humanas y variaciones ambientales permitirá al modelo adaptarse mejor a condiciones diversas y complejas. Esto incluye la incorporación de diferentes edades, géneros y contextos culturales en los datos de entrenamiento.

7.2.3 Evaluación del Modelo

Para obtener una evaluación más integral del desempeño del modelo, sería beneficioso implementar métricas de rendimiento adicionales que no solo ponderen la precisión global, sino que también midan la habilidad del modelo para reconocer correctamente acciones menos frecuentes o más sutiles. Estas métricas proporcionarían una evaluación más detallada y significativa del desempeño del modelo.

7.2.4 Actualización de Hardware

Considerar la actualización del hardware utilizado para procesar y analizar las imágenes es esencial. La adopción de hardware más avanzado y capaz de manejar imágenes de alta resolución sin comprometer la velocidad de procesamiento permitirá capturar detalles finos y sutilezas en la postura y el movimiento humano, cruciales para la precisión en el reconocimiento de acciones. La inversión en GPUs más potentes o sistemas especializados en procesamiento de imágenes es fundamental para manejar eficientemente el aumento en la carga computacional. Además, explorar frameworks de software alternativos que estén optimizados para operar con estos nuevos componentes de hardware podría ofrecer mejoras adicionales en términos de eficiencia computacional y facilitar la implementación del modelo en diferentes plataformas.

En resumen, aunque este estudio ha logrado avances significativos en el reconocimiento de actividades y acciones en vídeo, las mejoras continuas en la metodología y los recursos disponibles son esenciales para alcanzar niveles de precisión y generalización óptimos en aplicaciones prácticas. La implementación de estas mejoras y exploraciones propuestas permitirá seguir avanzando en el desarrollo de modelos de deep learning más efectivos y robustos para el reconocimiento de acciones humanas.

Bibliografía

Dai, R., Minciullo, L., Garattoni, L., Francesca, G., y Bremond, F. (2019). Self-attention temporal convolutional network for long-term daily living activity detection. En *Proceedings of the 15th ieee international conference on advanced video and signal based surveillance (avss)* (pp. 1–6). doi: 10.1109/AVSS.2019.8909841

DataPort, I. (2023a). *Falld - comprehensive dataset for human falls and activities of daily living.* Descargado de `https://ieee-dataport.org/open-access/fallalld-comprehensive-dataset-human-falls-and-activities-daily-living`

DataPort, I. (2023b). *Simulated falls and daily living activities dataset.* Descargado de `https://ieee-dataport.org/documents/simulated-falls-and-daily-living-activities-dataset`

Google. (2024a). *Mediapipe: A framework for building perception pipelines.* Descargado de `https://developers.google.com/mediapipe`

Google. (2024b). *Movenet: A fast and accurate model for pose estimation.* Descargado de `https://www.tensorflow.org/hub/tutorials/movenet?hl=es-419`

INRIA. (2023). *Toyota smarthome: Real-world untrimmed videos for activity detection.* Descargado de `https://project.inria.fr/toyotasmarthome/`

iWatch, S. (2023). *Activities of daily living dataset.* Descargado de `https://ieee-dataport.org/documents/activities-daily-living`

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., … Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. En *Proceedings of the 22nd acm international conference on multimedia* (pp. 675–678).

Labs, V. (2023). *Dahlia: High semantic activity dataset for online recognition.* Descargado de `https://www.v7labs.com/open-datasets/dahlia`

Mlinac, M. E., y Feng, M. C. (2021). Assessment of activities of daily living, self-care, and independence. *Gerontology and Geriatrics Studies*, *45*(2), 234–249.

Paraschiakos, S., Cachucho, R., Moed, M., van Heemst, D., Mooijaart, S., Slagboom, E. P., … Beekman, M. (2020). Activity recognition using wearable sensors for tracking the elderly. *User Modeling and User-Adapted Interaction*, *30*, 567–605. doi: 10.1007/s11257-020-09268-2

Paraschiakos, S., de Sá, C. R., Okai, J., Slagboom, P. E., Beekman, M., y Knobbe, A. (2022). A recurrent neural network architecture to model physical activity energy expenditure in

older people. *Data Mining and Knowledge Discovery*, *36*, 477–512. doi: 10.1007/s10618-021-00817-w

Vaquette, G., Orcesi, A., Lucat, L., y Achard, C. (2024). The daily home life activity dataset: A high semantic activity dataset for online recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. (https://ssrn.com/abstract=3595738)

Yu, B. X., Liu, Y., Zhang, X., Zhong, S.-h., y Chan, K. C. (2023). Mmnet: A model-based multimodal network for human action recognition in rgb-d videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(3), 3522–3537. doi: 10.1109/TPAMI.2022.3177813

Zhang, C., y Czarnuch, S. (2023). Point cloud completion in challenging indoor scenarios with human motion. *Frontiers in Robotics and AI*, *10*, 1184614. doi: 10.3389/frobt.2023.1184614

Zhu, H., Samtani, S., Brown, R. A., y Chen, H. (2023). A deep learning approach for recognizing activity of daily living (adl) for senior care: Exploiting interaction dependency and temporal patterns. *Journal of Biomedical Informatics*, *127*, 104019. doi: 10.1016/j.jbi.2022.104019