



UNIVERSIDAD DE GRANADA

TRABAJO FIN DE MÁSTER

MÁSTER EN CIENCIA DE DATOS E INGENIERÍA DE COMPUTADORES (DATCOM)

Aprendizaje profundo para el análisis de comportamientos en videovigilancia

Deep Learning for crowd behavior analysis in videosurveillance

Autor

Francisco Luque Sánchez

Directores

Francisco Herrera Triguero
Siham Tabik



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

—
Granada, 10 de septiembre de 2020

Aprendizaje profundo para el análisis de comportamientos en videovigilancia

Francisco Luque Sánchez

Palabras clave:

Aprendizaje profundo, análisis de multitudes, detección de anomalías, videovigilancia

Resumen

En las últimas décadas se ha producido un crecimiento poblacional sin precedentes en todas las partes del mundo, y las tasas de criminalidad y terrorismo se han disparado en muchos territorios. Esto ha provocado que la videovigilancia se convierta en una herramienta prioritaria a nivel mundial. El número de cámaras de seguridad instaladas tanto en ámbito público como privado ha crecido significativamente, y con ello la dificultad de gestionar la información recogida por las mismas de forma manual. Aparece, por tanto, la necesidad de automatizar este proceso, utilizando para ello modelos inteligentes capaces de extraer información de las secuencias de vídeo recogidas por las cámaras.

Los avances en aprendizaje profundo de los últimos años han permitido que los resultados obtenidos sobre esta área de investigación mejoren considerablemente. Los modelos actuales son capaces de extraer información más compleja, y trabajar en entornos de mayor dificultad, especialmente en cuanto a densidad de individuos se refiere. A pesar de ello, el estudio de esta tarea es relativamente reciente, por lo que no está correctamente estructurada, y resulta difícil comparar los trabajos propuestos.

Dado el contexto anterior, este trabajo trata de resolver varias tareas relacionadas con el tratamiento automático de fuentes de videovigilancia, en particular con el análisis de comportamientos de multitudes. Por un lado, se realiza un estudio teórico de la temática, con una propuesta taxonómica que permite agrupar los distintos trabajos siguiendo una secuencia de tareas. Esta organización sitúa las distintas subtareas consideradas dentro de la temática en distintos pasos de la secuencia, de forma que los resultados de los pasos posteriores se ven fuertemente influenciados por los obtenidos en los pasos previos.

Además de la propuesta taxonómica, en el estudio teórico se hace una revisión exhaustiva de la literatura que utiliza modelos de aprendizaje profundo para resolver el problema de la detección de anomalías en multitudes. Para esta subtarea, se analizan los principales conjuntos de datos disponibles públicamente, y se estudian los trabajos del estado del arte, agrupando los mismos por el tipo concreto de anomalía que tratan de identificar.

En el apartado práctico del trabajo, se estudia el uso de características espacio-temporales para la detección de acciones anómalas en vídeo. Para llevar a cabo dicho estudio, se establece como punto de partida un modelo de detección de anomalías basado en un extractor de características convolucional en tres dimensiones. Nuestra propuesta, en lugar de utilizar un extractor exclusivamente convolucional, aprovecha la potencia de las redes neuronales convolucionales 2D para el análisis de los fotogramas por separado, extrayendo información espacial, y la capacidad de las redes neuronales recurrentes para extraer información temporal de la secuencia de características de los fotogramas consecutivos. Dicho extractor de características es más complejo que la propuesta original, y conserva mejor la estructura temporal del vídeo, lo cual permite la extracción de información de mayor calidad. El código desarrollado para el experimentación se encuentra disponible en el repositorio <https://github.com/fluque1995/tfm-anomaly-detection>.

Los resultados obtenidos del estudio sugieren que nuestro modelo tiene mejor capacidad de clasificación que el modelo original, incluso a pesar de estar entrenado en un conjunto de datos de tamaño 1000 veces menor que el extractor de características de partida. Este hecho nos permite concluir que nuestra propuesta es de mayor calidad que el modelo de partida, validando nuestra hipótesis inicial.

Deep Learning for crowd behavior analysis in videosurveillance

Francisco Luque Sánchez

Keywords:

Deep learning, crowd analysis, anomaly detection, video-surveillance

Abstract

Last decades have experimented and unprecedented growth in population all around the globe. Crime and terrorism rates are also increasing rapidly. This two issues have converted video-surveillance into a fundamental tool worldwide. The number of security cameras installed both at public and private environments is huge nowadays, and thus processing the information retrieved by those cameras is getting harder everyday. Then, there is a need of automation of that process, using intelligent models capable of extracting information from video sources.

Advances in deep learning produced in the last years have improved the results obtained in this research area by a large margin. Current models are able to extract complex information automatically, and to work in more difficult environments, specially in terms of density of individuals. Despite this recent advances, the area is still relatively modern, and consequently is not properly structured. Due to this fact, comparing works that tackle different sub-tasks within this area is usually difficult.

In the previous context, this work tries to solve different tasks related to the automatic treatment of video-surveillance sources. In particular, we will focus our efforts in the analysis of crowd behaviors from video-surveillance sources. The work is divided in two parts. In the first part, which is the theoretical study, a sequential taxonomy is proposed. This taxonomy following a sequence allows to organize the different sub-tasks inside the topic as stages of a pipeline. The results of the latter stages of the pipeline strongly rely on the previous ones, and thus its results are heavily influenced by the first models.

Apart from the proposed taxonomy, an in-depth review of the state-of-the-art is conducted. Particularly, we center our study in the works that use deep learning models to solve the crowd anomaly detection problem. Inside this topic, we analyze the main public datasets and proposals, organizing

them depending on the specific type of anomaly to be detected.

Finally, in the experimental part of this work, the use of spatio-temporal features for action anomaly detection is studied. To this end, one of the retrieved works is deeply analyzed and set as baseline for comparison. After the complete study, a new model for action anomaly detection is proposed. In particular, the original model employs a 3D convolutional feature extractor, which processes batches of consecutive frames. In our approach, the feature extractor proposed is a combination of 2D CNNs for spatial feature extraction, processing frames independently, together with a recurrent neural network, which learns to process the sequence of features from consecutive frames in the video. Our hypothesis defends that the combination of convolutions and recurrent networks better preserves the semantic structure of information in the video, and thus the obtained model extracts more meaningful information.

Our experimentation suggests that our model outperforms the classification capability of the original model, despite being pretrained on a much smaller dataset. In particular, the original feature extractor is trained on a set 1000 times bigger than ours. This fact allows us to conclude that our proposal is better than the original one in terms of classification capability, validating our hypothesis.

The code developed for experimentation, together with the instructions to replicate the experiments, can be found in the repository <https://github.com/fluque1995/tfm-anomaly-detection>.

Yo, **Francisco Luque Sánchez**, alumno del Máster Universitario Oficial en Ciencia de Datos e Ingeniería de Computadores de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 31008316S, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Máster en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.



Fdo: Francisco Luque Sánchez

Granada, 10 de septiembre de 2020

D. Francisco Herrera Triguero y D.^a Siham Tabik, profesores del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado *Aprendizaje profundo para el análisis de comportamientos en videovigilancia*, ha sido realizado bajo su supervisión por **Francisco Luque Sánchez**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 10 de septiembre de 2020

Los directores:



Francisco Herrera Triguero



Siham Tabik

Índice general

Índice general	7
1 Introducción	9
1.1. Objetivos del trabajo	11
2 Estado del arte en análisis de multitudes en videovigilancia	13
2.1. Propuesta taxonómica	13
2.1.1. Categorizaciones previas	13
2.1.2. Taxonomía propuesta	14
2.2. Conjuntos de datos públicos para detección de anomalías en multitudes	20
2.2.1. Conjuntos de datos para la detección de movimientos anómalos	20
2.2.2. Conjuntos de datos para detección de acciones anómalas	23
2.3. Métricas de evaluación utilizadas en la detección de anomalías	25
2.3.1. Métricas clásicas	25
2.3.2. Métricas a nivel de fotograma y a nivel de píxel	27
2.4. Revisión de la literatura	27
2.4.1. Detección de movimiento y apariencia anómala	28
2.4.2. Detección de acciones anómalas	34
2.4.3. Detección de posiciones anómalas	38
3 Características espacio-temporales para la detección de acciones anómalas	39
3.1. Conjunto de datos empleado: UCF-Crime Dataset	39
3.2. Modelo original	43
3.2.1. Extractor de características: C3D	43
3.2.2. Aprendizaje multi-instancia	46
3.2.3. Modelo original completo	48
3.3. Propuesta de mejora del modelo	49
3.3.1. Extractor de características: Xception-LSTM	50
3.3.2. Arquitectura detectora de anomalías completa	54

4 Experimentación y resultados obtenidos	55
4.1. Aspectos de implementación	55
4.2. Evaluación de los modelos	57
4.3. Reproducción de la experimentación original	59
4.3.1. Extracción de características	59
4.3.2. Entrenamiento del clasificador	60
4.3.3. Inferencia sobre el conjunto de test	61
4.4. Experimentación propia	61
4.4.1. Entrenamiento del extractor Xception-LSTM	61
4.4.2. Extracción de características	66
4.4.3. Entrenamiento del clasificador	66
4.5. Resultados obtenidos	67
5 Conclusiones y trabajo futuro	76
5.1. Conclusiones	76
5.2. Trabajo futuro	78
5.3. Publicaciones asociadas	80
Bibliografía	82

1 Introducción

En las últimas décadas se ha experimentado un crecimiento poblacional sin precedentes alrededor de todo el mundo, con el consecuente aumento de las aglomeraciones de personas, las cuales llegan a involucrar a miles de individuos. Además, las tasas de criminalidad y terrorismo se han disparado de forma similar. La combinación de estos hechos ha provocado que la videovigilancia masiva se convierta en una herramienta prioritaria. El número de cámaras de vigilancia instaladas en el mundo, tanto dentro del ámbito público como en el privado, se ha multiplicado en los últimos años. El desarrollo tecnológico, además, está produciendo una importante mejora en la calidad de los vídeos que se recopilan, a cambio de un aumento importante en el volumen de información almacenada. Aparece, por tanto, la necesidad de procesar una gran cantidad de información en forma de archivos de vídeo.

Históricamente, dicho procesamiento se ha realizado de forma manual, consumiendo una gran cantidad de recursos humanos. Hoy en día, la velocidad a la que se genera dicha información, y el volumen tan abismal que se genera diariamente, hace casi imposible la gestión manual de esta información de forma exhaustiva y adecuada. Además, esta información debe procesarse en tiempo real en la medida de lo posible, ya que la respuesta rápida en situaciones de emergencia es crucial para reducir los efectos de una posible catástrofe. Esto ha hecho que los métodos clásicos de supervisión humana queden paulatinamente obsoletos, y aparezca la necesidad de automatizar el proceso. En este contexto aparece el concepto de la videovigilancia automática.

La videovigilancia automática es una rama de investigación cuyo objetivo es el análisis de múltiples fuentes de vídeo en tiempo real, para la extracción automática de información relevante relacionada con el comportamiento de los individuos [1]. Esta área de investigación aúna dos grandes campos de trabajo dentro del aprendizaje automático; la visión por computador y el análisis de series temporales. Dado que el tipo de dato más común dentro

de este contexto son las secuencias de vídeo, por un lado se ha de extraer información de cada uno de los fotogramas, y por otro información temporal derivada de la secuencia de dichos fotogramas.

El auge del aprendizaje profundo, además, ha supuesto un avance muy importante en el desarrollo de modelos en este contexto. Según [2], existen cuatro áreas principales en el contexto de la videovigilancia automática:

1. Detección y seguimiento de individuos
2. Recuento y estimación de densidad de individuos
3. Análisis y clasificación de comportamientos
4. Detección de comportamientos anómalos

Las tareas 1 y 2 han sido ampliamente estudiadas y los modelos de aprendizaje clásico son suficientes para la obtención de resultados de calidad. No obstante, en las tareas 3 y 4 los resultados eran muy limitados. La aparición de los modelos de aprendizaje profundo y el aumento de la capacidad de cálculo ha supuesto un avance importante para todas las áreas. En los dos primeros casos, ha permitido que la densidad de individuos presentes en la imagen sea más alta antes de que se produzca una pérdida de rendimiento, y en los dos últimos casos ha provocado una mejoría muy notable, ya que la complejidad de la información que estos modelos son capaces de extraer es notablemente más alta que la extraída por los modelos clásicos.

Esta mejora tan significativa ha provocado que en los últimos años aparezcan una gran cantidad de trabajos que resuelven alguno de los problemas relacionados con la videovigilancia automática aplicando modelos de redes neuronales. No obstante, estos trabajos aparecen dispersos, y es difícil establecer una comparativa sobre ellos. Esta problemática es especialmente relevante cuando se tratan de desarrollar nuevos modelos, ya que es difícil recopilar el conocimiento previo sobre la temática. Esta dispersión radica en varios factores:

1. No existe un consenso claro sobre las tareas que deben abordarse dentro de esta área de investigación.
2. No hay una taxonomía clara para organizar los distintos trabajos previamente desarrollados.

3. No existe una recopilación de trabajos que afronten esta problemática desde la perspectiva del aprendizaje profundo.

En particular, nuestra propuesta de trabajo se engloba dentro del proyecto de investigación *AI_MARS-DeepLABD: Artificial Intelligence system for Monitoring, Alert and Response for Security in events. Deep Learning for Abnormal Behavior Detection*, el cual pretende diseñar e implementar sistemas de aprendizaje profundo para la detección de comportamientos anómalos, y por tanto, enfocaremos el trabajo en esa dirección. Esto hace que nos centremos especialmente en la última de las tareas. Es la más novedosa de las áreas listadas anteriormente, y esto hace que exista una especial incertidumbre alrededor de la misma. En particular, resulta muy difícil establecer una organización clara para los trabajos que afrontan esta problemática, porque el concepto de comportamiento anómalo puede incluir definiciones muy diversas. Por ejemplo, podemos considerar como anómalo la presencia de una persona en un área restringida, una multitud corriendo despavorida, o un pequeño grupo de personas que inicia una pelea por la calle. Claramente, la fuente de la anomalía en los tres casos es completamente distinta, y difícilmente comparable. Esto provoca que la comparativa entre modelos sea compleja. Además, el número de conjuntos de datos públicos que permitan establecer comparaciones entre los modelos es relativamente escaso, y un gran número de trabajos utilizan sus propios conjuntos de datos diseñados específicamente para el problema que tratan de resolver.

1.1. Objetivos del trabajo

Dado el contexto previo, los objetivos de este trabajo tratan de cubrir un estudio profundo del área de la videovigilancia automática, en particular centrado en la detección de comportamientos anómalos en vídeo. Los objetivos concretos que se han planteado para el trabajo son los siguientes:

1. Proponer una taxonomía para la organización de los trabajos que afrontan el problema del análisis de multitudes en videovigilancia.
2. Revisar detalladamente los trabajos propuestos dentro del área de la detección de comportamientos anómalos utilizando aprendizaje profundo.
3. Estudiar la extracción de características en vídeo utilizando modelos de aprendizaje profundo. En concreto, se estudia un modelo del estado del

arte en la detección de anomalías en videovigilancia, y se propone una mejora basada en un extracto de características profundo de mayor potencia.

El resto del trabajo se estructura de la siguiente manera. En el capítulo 2 se expone el estudio teórico del trabajo. En él, se propone una taxonomía en etapas que permite agrupar los trabajos hasta el momento dentro de cuatro etapas, en las que cada una recae en los resultados de las anteriores. Además, se estudian los principales conjuntos de datos disponibles públicamente y las métricas que se utilizan para evaluar la calidad de los modelos dentro de esta área de conocimiento. Finalmente, se resumen los principales trabajos que utilizan aprendizaje profundo para resolver el problema de la detección de anomalías en multitudes, estableciendo una división de los mismos en función de la taxonomía previa.

En el capítulo 3 se llevará a cabo el análisis del uso de características espacio-temporales para la detección de acciones anómalas. Concretamente, tomaremos el trabajo propuesto en [3] y trataremos de mejorar sus resultados empleando un extracto de características más potente. En dicho trabajo, proponen una red neuronal convolucional en tres dimensiones como extracto de características en vídeo. A pesar de ser un modelo relativamente bueno para capturar características tanto temporales como espaciales, creemos que las características temporales se ven infrarrepresentadas. Nuestra hipótesis expone que el uso de un extracto de características más potente, que combine la capacidad de trabajar con imágenes de las redes convolucionales 2D con la capacidad de analizar series temporales de las redes recurrentes, obtendrá mejores resultados en el problema tratado. En este capítulo se realiza un estudio teórico del modelo original y se realiza nuestra propuesta de mejora.

A continuación, en el capítulo 4 se detalla la experimentación realizada y los resultados obtenidos. Finalmente, en 5 se exponen las conclusiones derivadas del estudio y posibles estudios futuros.

Estado del arte en análisis de multitudes en videovigilancia

En este capítulo se va a realizar un análisis sobre el estado del arte en el análisis de comportamientos en videovigilancia. El contenido de este capítulo consiste en un resumen de las ideas fundamentales que aparecen en el artículo “Revisiting crowd behaviour analysis through deep learning: Taxonomy, anomaly detection, crowd emotions, datasets, opportunities and prospects”. Este artículo se ha redactado tras una investigación durante este curso académico, y está aceptado para publicación en la revista *Information Fusion*.

2.1. Propuesta taxonómica

En esta sección se introduce una propuesta taxonómica para la organización de los trabajos que abordan el análisis de comportamientos en multitudes. Esta taxonomía aparece tras el análisis de las propuestas anteriores, en las que se observa una carencia de estructura que organice las distintas tareas que componen el área.

2.1.1. Categorizaciones previas

La categorización más utilizada para la organización de los trabajos que se engloban dentro de esta área es la propuesta en [2]. Dicha estructuración divide los trabajos que se engloban dentro del análisis de comportamientos en multitudes en cuatro subproblemas:

- Clasificación de comportamientos: Esta tarea consiste en identificar y clasificar los comportamientos presentes en el vídeo dentro de un conjunto de comportamientos conocidos de antemano.
- Conteo de individuos: Esta tarea consiste en estimar el número de individuos presentes en cada fotograma.
- Detección y seguimiento de individuos: En esta tarea se engloban los trabajos cuyo objetivo es el seguimiento de la trayectoria de los individuos en el vídeo.

- Detección de comportamientos anómalos: En este caso, se trata de identificar cuándo se produce un comportamiento anormal o desconocido en la secuencia de vídeo.

Dicha taxonomía ha sido utilizada como referencia a la hora de realizar estudios posteriores del estado del arte en la temática [4]. Esta categorización incluye las tareas principales dentro de este ámbito de estudio, pero sitúa todas las tareas al mismo nivel. No obstante, existe claramente una organización jerárquica entre las tareas listadas, y pueden organizarse las mismas siguiendo una secuencia. Por ejemplo, el conteo de individuos suele venir precedido por una etapa de detección de individuos en la escena. Además, la salida del conteo puede utilizarse en etapas posteriores para detectar situaciones anómalas, como una excesiva congestión en una aglomeración de personas. La taxonomía que se propone establece una jerarquía entre las diferentes tareas, al tiempo que mantiene las aportaciones de los trabajos anteriores como etapas dentro de la jerarquía.

2.1.2. Taxonomía propuesta

Con la taxonomía propuesta a continuación se pretende establecer una conexión entre las distintas subtareas que aparecen dentro de esta área de conocimiento. La propuesta se vertebral en torno a dos ejes complementarios:

- Por un lado, hay dos formas principales de afrontar el análisis de comportamientos en multitudes, en función de la relación que se establece entre los individuos aislados y la multitud que forman. Esta distinción se traduce en los enfoques microscópico y macroscópico, que se explicarán más adelante.
- Por otro lado, independientemente del enfoque seleccionado, las diferentes tareas se organizan en cuatro etapas. Usualmente, las etapas posteriores suelen tener una fuerte dependencia en las etapas previas.

En la figura 2.1 puede apreciarse visualmente la organización anterior en dos ejes. Dicha organización muestra la interdependencia entre las dos organizaciones anteriores. Todas las etapas de la jerarquía pueden llevarse a cabo utilizando cualquiera de los dos enfoques. No obstante, la elección del enfoque microscópico o macroscópico producirá ligeras diferencias en la solución obtenida.

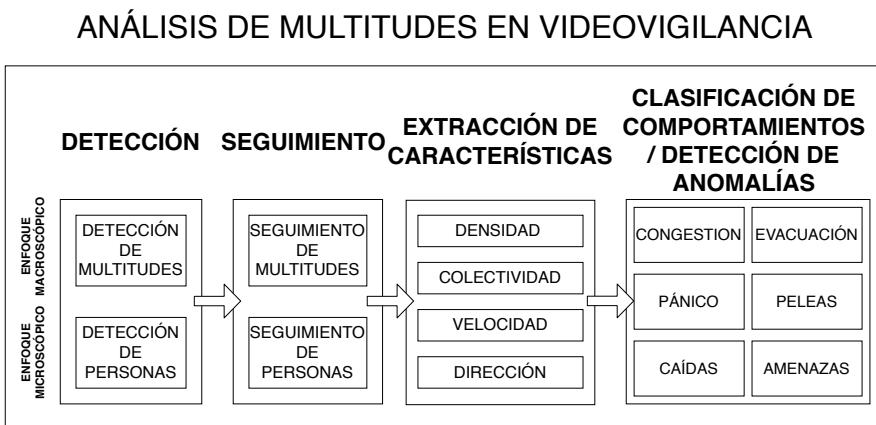


Figura 2.1: Propuesta taxonómica. En ella puede verse la organización de los trabajos en cuatro etapas consecutivas

Enfoque macroscópico contra enfoque microscópico

La primera distinción distribuye los trabajos en función de cómo se consideran los individuos en relación con la multitud a la que pertenecen. Como hemos dicho anteriormente, esta categorización no forma parte de la jerarquía, sino que es una clasificación complementaria que influye en gran medida a las distintas etapas. Los dos enfoques principales son los que siguen:

- Enfoque microscópico: En este caso, se trata a la multitud como una colección de individuos. Las distintas personas que aparecen en el vídeo se estudian individualmente, y posteriormente se relaciona la información obtenida de las mismas para inferir la información a nivel de multitud.
- Enfoque macroscópico: Estos son enfoques holísticos, en los que la multitud es tratada como un ente único y completo, sin necesidad de estudiar individualmente a las personas que la componen.

Normalmente, los enfoques microscópicos tienden a arrojar mejores resultados cuando los individuos pueden ser estudiados de forma aislada correctamente. Esto es, en entornos en los que los individuos son claramente visibles, las occlusiones son pequeñas, y la densidad de personas es baja. Por otra parte, cuando la densidad de individuos aumenta, la calidad de los algoritmos de seguimiento se degrada significativamente. En estas circunstancias, los enfoques macroscópicos suelen ser más adecuados, ya que los individuos específicos dejan de ser el objetivo, y se pasa a estudiar la multitud de forma completa.

Jerarquía de tareas

Independientemente del enfoque escogido, la jerarquía de tareas incluye las cuatro fases indicadas en la figura 2.1:

1. Detección: El objetivo de esta fase es localizar la posición de los individuos (en los enfoques microscópicos) o las multitudes (en los macroscópicos) en cada fotograma. Esta etapa ha sido ampliamente estudiada con anterioridad, y existen diversos modelos con gran precisión y rendimiento [5]
2. Seguimiento de individuos: Se trata de identificar únicamente a las personas o multitudes durante una secuencia de fotogramas consecutivos. En el caso de multitudes, se suelen calcular también los flujos de movimiento dominantes [6]. De nuevo, es un área ampliamente estudiada [7].
3. Extracción de características: El objetivo de esta etapa consiste en el cómputo de una serie de métricas que describan la dinámica de la multitud o los individuos en escena. Algunos ejemplos de estas características pueden ser la velocidad o dirección de las trayectorias, o la densidad de individuos.
4. Clasificación de comportamientos y detección de anomalías: Utilizando como base las características extraídas en la etapa anterior, este paso trata de reconocer comportamientos particulares o situaciones anómalas en la secuencia de vídeo. En función del paradigma de aprendizaje utilizado, existen dos posibles enfoques para esta etapa:
 - La clasificación de comportamientos engloba a los trabajos que utilizan un enfoque supervisado, los cuales tratan de asignar una clase conocida previamente al comportamiento a clasificar.
 - La detección de anomalías trata de identificar patrones anómalos utilizando un enfoque no supervisado para resolver el problema.

Las dos últimas etapas de la jerarquía van a ser analizadas en mayor profundidad en las secciones posteriores. Las dos primeras secciones han sido ampliamente estudiadas con anterioridad, por lo que excluiremos su análisis en este trabajo.

Características relevantes para el estudio de multitudes

Tras las etapas de detección y seguimiento de individuos o multitudes en la escena, suele encontrarse una etapa de extracción de características. A pesar de que la información extraída puede ser de muy diversa índole, se han identificado como relevantes las siguientes características, debido a que un amplio número de trabajos utilizan alguna de ellas en su funcionamiento. Dichas características son:

- Velocidad: Mide la velocidad media a la que se mueven los individuos o la multitud en la escena.
- Dirección: En el enfoque macroscópico, número de direcciones principales de movimiento presentes en la multitud. En el microscópico, se puede extraer la dirección de cada uno de los individuos en escena.
- Densidad o conteo: Mide el número de individuos presentes en la imagen, o la proximidad que existe entre ellos cuando se estudian multitudes completas.
- Colectividad o cohesión: Esta característica mide la fuerza con la que se relacionan los individuos que conforman la multitud. Cuando una persona forma parte de un grupo grande, en lugar de comportarse individualmente, tiende a imitar el comportamiento de los individuos que le rodean, asimilando su dirección y velocidad. De esta forma, aparecen estructuras coherentes debidas al movimiento. La colectividad trata de medir la estabilidad de dichas estructuras, y depende de múltiples factores.

Adicionalmente, la etapa de extracción de características puede llevarse a cabo con un modelo de aprendizaje profundo, que aprenda a seleccionar las características más relevantes para la tarea que se trata de resolver. Este enfoque en particular es el que utilizaremos en la experimentación práctica del trabajo, donde un modelo espacio-temporal será el encargado de aprender las características relevantes.

Clasificación de comportamientos y detección de anomalías

Las características extraídas en la etapa anterior tienen que ser resumidas para obtener información relevante sobre la multitud estudiada. Como hemos explicado anteriormente, existen dos enfoques principales para abordar esta etapa. Por un lado, tenemos la clasificación de comportamientos, en la que los modelos se entrena utilizando un paradigma supervisado sobre las

características extraídas, y la detección de anomalías, en la que el aprendizaje se realiza desde el punto de vista no supervisado. La monitorización de las características a lo largo del tiempo permite la detección de comportamientos anómalos en la multitud, especialmente cuando se producen cambios bruscos en dichas métricas. Por ejemplo, un aumento repentino de la velocidad media de los individuos en la imagen puede ser un indicador de alerta, ya que puede estar ocurriendo una estampida, o una congestión puede ser expresada en términos de alta densidad y baja velocidad.

Cuando el problema se resuelve desde el punto de vista de la detección de anomalías, otra posible clasificación aparece en función de la fuente que produce la anomalía. El motivo por el que ocurre el evento anómalo puede ser de diversa índole, y por lo tanto la forma de afrontar el problema puede ser ligeramente distinta. Se han identificado cuatro principales fuentes de anomalía en la revisión de la literatura:

- Posición anómala: Esta anomalía ocurre cuando un objeto tiene una posición atípica en la escena. Ocurre normalmente cuando un individuo se en una zona no autorizada, por ejemplo. Es la anomalía más simple de las estudiadas, ya que usualmente se resuelve con una fase de detección de personas y cálculo de solapamiento entre la detección y la zona restringida.
- Movimiento anómalo: En este caso, la anomalía ocurre porque se detecta una trayectoria inesperada, bien por un único individuo o por un grupo. Hay dos posibles fuentes de irregularidad en este caso: la velocidad, cuando alguien se mueve más rápido o más lento que la gente que lo rodea, y la dirección, cuando hay flujos de movimiento predominantes y una trayectoria se desvíe significativamente de los mismos.
- Apariencia anómala: Esta anomalía se produce cuando un objeto no identificado aparece en la escena. Un ejemplo típico de esta anomalía es la presencia de vehículos en zonas peatonales.
- Acción anómala: Es la anomalía más difícil de detectar dentro de las comentadas. Conlleva el aprendizaje de los patrones de comportamiento habituales para una escena determinada, y la detección de patrones poco comunes.

En la práctica, muchas de estas anomalías aparecen combinadas. Por ejemplo, es típico que la anomalía de movimiento y la de apariencia aparezcan

simultáneamente en los conjuntos de datos. En particular, en el UCSD Pedestrian Dataset [8], el más utilizado en la temática, aparecen tanto ejemplos de movimiento anómalo como vehículos no autorizados en una calzada peatonal. Pueden observarse distintos ejemplos de anomalía presentes en conjuntos de datos públicos en la figura 2.2.



(a) Ejemplo de anomalía en el UMN Dataset [9]. En el vídeo, una multitud poco estructurada se desplaza por la imagen. En cierto momento, el movimiento pasa a ser acelerado por una situación de pánico, así que se produce una anomalía de movimiento



(b) Ejemplo de anomalía en el UCSD Pedestrian Dataset [8]. Dado que no se permite la circulación de vehículos por esa acera, la presencia del coche es una anomalía de apariencia.



(c) Ejemplo de acción anómala en el BOSS Dataset [10]. En este caso, el hombre roba el teléfono a la mujer mientras está hablando.



(d) Ejemplo de anomalía de movimiento en el conjunto de datos CUHK Avenue Dataset [11]. En este conjunto de datos, el plano de movimiento normal es paralelo al plano de la cámara, por lo que un movimiento en dirección perpendicular se considera un patrón anómalo de movimiento.

Figura 2.2: Ejemplos de anomalías en los distintos conjuntos de datos.

2.2. Conjuntos de datos públicos para detección de anomalías en multitudes

Debido a la diversidad que existe en la tarea de detección de comportamientos anómalos en multitudes, existe una gran cantidad de conjuntos de datos distintos disponibles públicamente. Cada conjunto de datos se centra en una o varias de las categorías de anomalía que hemos listado anteriormente. En esta sección se detallarán los principales conjuntos de datos en función de la anomalía que presenten. En la tabla 2.1 se resumen las principales características de todos los conjuntos de datos que se han estudiado.

2.2.1. Conjuntos de datos para la detección de movimientos anómalos

Los conjuntos de datos listados en esta sección están diseñados para presentar patrones anómalos de movimiento. Estas anomalías están definidas normalmente por velocidades o direcciones que se desvían de los flujos de movimiento normales de la escena. Además, la presencia de elementos no autorizados es común en estos conjuntos, como vehículos o bicicletas en zonas de tránsito de peatones. Esto hace que normalmente se considere simultáneamente el problema de detección de movimientos y apariencias anómalas. Los conjuntos de datos más utilizados en este contexto son los siguientes:

UCSD Pedestrian Dataset Los conjuntos *UCSD anomaly detection datasets*¹ [8] son los conjuntos de datos más populares en la literatura. Hay dos conjuntos de datos diferentes, llamados Peds1 y Peds2. Peds1 contiene 34 vídeos de entrenamiento y 36 de test y Peds2, 16 de entrenamiento y 12 de test. Cada vídeo dura aproximadamente 200 fotogramas (20 segundos), y están grabados a una resolución de 158x238 píxeles. La principal diferencia entre los dos conjuntos es la dirección en la que se mueven los peatones. En Peds1, la gente se mueve en dirección perpendicular al plano de la cámara, mientras que en Peds2 la dirección predominante es paralela. En los conjuntos de datos de entrenamiento no aparecen trayectorias anómalas, con el fin de mostrar los patrones de movimiento que se consideran usuales. En el conjunto de test, un fotograma se considera anómalo si hay algún elemento extraño en la imagen (bicicletas, monopatines, etc), o si un viandante presenta un patrón extraño de movimiento (velocidad anormalmente alta, o dirección no

¹UCSD dataset disponible en: <http://www.svcl.ucsd.edu/projects/anomaly/dataset.htm>

Tabla 2.1: Conjuntos de datos para detección de anomalías en multitudes.

Conjunto de datos	Nº fotogramas			Nº eventos anómalos	Tipo de anomalía	Descripción de las anomalías
	Total	Entrenamiento	Test			
UCSD Peds 1	14000	6800	7200	40	Movimiento + apariencia	Direcciones extrañas, velocidad, vehículos no autorizados
UCSD Peds 2	4560	2550	2010	12	Movimiento + apariencia	Direcciones extrañas, velocidad, vehículos no autorizados
CUHK Avenue	30652	15328	15324	47	Movimiento + apariencia	Direcciones extrañas, velocidad, vehículos no autorizados
ShangaiTech Campus	317398	274515	42883	130	Movimiento + apariencia	Direcciones y velocidades anómalas, merodeo
UMN Dataset	7725	-	-	3	Movimiento	Multitud despavorida instantáneamente
BEHAVE Interactions	225019	-	-	14 ^a	Acción	Peleas, persecuciones
CAVIAR	26402	-	-	11 ^b	Acción	Objetos abandonados, peleas, caídas
BOSS	48624	-	-	10	Acción	Peleas, robos, caídas
UT Interactions	41373	-	-	48	Acción	Puñetazos, patadas, empujones
UCF Crime	13M	-	-	-	Acción	Comportamientos peligrosos ^c
Peliculas	4991	-	-	100 ^d	Acción	Violencia
Hockey Fights	41056	-	-	500 ^d	Acción	Violencia

^a Sólo uno de los vídeos está etiquetado.^b La etiquetación permite también la clasificación de comportamientos.^c Vídeos normales junto con 13 clases de comportamientos anómalos.^d Vídeos de corta duración divididos en violencia y no violencia.

predominante. En total, unos 3400 fotogramas son anómalos, frente a unos 5500 que son normales. Hay dos formas distintas de marcar las anomalías:

- Para cada vídeo hay una marca binaria por fotograma, indicando si el mismo es anómalo o no (anotaciones a nivel de fotograma)
- Adicionalmente, para 10 vídeos de Peds1, se adjuntan máscaras que indican la posición exacta de la anomalía dentro del fotograma (anotaciones a nivel de píxel).

A pesar de ser el conjunto de datos más empleado en la literatura, resulta difícil comparar los trabajos que lo utilizan, debido a que no existe una tabla de resultados centralizada. La mayoría de trabajos que utilizan este dataset resuelven simultáneamente la detección de movimientos y apariencias anómalas, dado que en las anotaciones del conjunto de datos no se hace distinción en la fuente de la anomalía.

CUHK Avenue Dataset El conjunto de datos CUHK Avenue² [11] está compuesto por 16 vídeos de entrenamiento (15328 fotogramas) y 21 de test (15324 fotogramas). De nuevo, se consideran trayectorias normales aquellas que se mueven en un plano paralelo a la cámara, mientras que la gente moviéndose en otras direcciones, con patrones de movimiento extraños o los vehículos se consideran anomalías. En este caso, las anotaciones marcan las anomalías con un rectángulo, y el criterio de evaluación es la métrica IoU (*intersection over union*, se explicará más adelante) entre la detección y la anotación.

UMN dataset El conjunto de datos UMN³ es un conjunto de datos sintético compuesto por tres escenas, con una longitud total ligeramente superior a los 4 minutos (7725 fotogramas). En cada uno de los vídeos se muestra una multitud poco cohesiva, que repentinamente empieza a correr en todas direcciones. El momento en el que se produce el cambio de velocidad se considera como inicio de la anomalía. Esta tarea puede verse alternativamente como detección de movimiento anómalo o como detección de acción anómala. Esto se debe a que la anomalía está producida por un cambio brusco de velocidad y dirección, características fuertemente relacionadas con las anomalías de movimiento, pero los

²El conjunto CUHK Avenue se puede descargar de <http://www.cse.cuhk.edu.hk/leojia/projects/detectabnormal/dataset.html>

³UMN está disponible en http://mha.cs.umn.edu/proj_events.shtml#crowd

patrones de movimiento en la escena están completamente desestructurados, en contraste con el movimiento estructurado de los conjuntos previos. Esta falta de estructura hace que no sea posible aprender los principales flujos de movimiento, por lo que las aproximaciones usuales al problema del movimiento anómalo son difícilmente aplicables aquí. El conjunto de datos Web [9] es una propuesta de mejora de este conjunto de datos, en el que se encuentran presentes multitudes más densas.

ShanghaiTech Campus Dataset El conjunto de datos ShanghaiTech Campus⁴ [12] está dividido en 330 vídeos de entrenamiento y 107 secuencias de test. Las escenas están tomadas en 13 localizaciones distintas dentro del campus de la universidad. Los eventos anómalos están provocados por objetos extraños en la escena, peatones con velocidades anómalas de movimiento (carreras y merodeos), y direcciones de movimiento poco comunes.

2.2.2. Conjuntos de datos para detección de acciones anómalas

La tarea a resolver en estos conjuntos de datos consiste en identificar cuándo una persona presente en la escena tiene un comportamiento anómalo. Normalmente, los comportamientos considerados anómalos son comportamientos incívicos, como robos, peleas, hurtos, etc. En algunos conjuntos, también se consideran eventos anómalos peligros para las personas como explosiones o incendios. Los conjuntos de datos más representativos para esta tarea son los siguientes:

BEHAVE Dataset El conjunto de datos BEHAVE Interactions⁵ [13] está formado por 4 secuencias, con una duración total de unas 2 horas. Sólo la primera de las secuencias está anotada oficialmente por los autores del conjunto de datos. Dicha secuencia está dividida en 8 fragmentos y anotada a nivel de fotograma. Los eventos anómalos que ocurren en este conjunto son peleas entre individuos y persecuciones.

CAVIAR Dataset CAVIAR Test Case Scenarios⁶ es un conjunto de escenas tomadas en dos localizaciones distintas: la entrada de un centro de in-

⁴ShanghaiTech Campus se puede descargar de https://svip-lab.github.io/dataset/campus_dataset.htm

⁵BEHAVE se puede descargar de <http://groups.inf.ed.ac.uk/vision/BEHAVEDATA/INTERACTIONS/>

⁶CAVIAR puede descargarse en <http://groups.inf.ed.ac.uk/vision/CAVIAR/CAVIARDATA1/>.

vestigación y un pasillo de un centro comercial. Hay varias secuencias de vídeo para cada una de las localizaciones. En cada vídeo se observa a una persona o pequeño grupo de personas realizando distintas acciones. Las anomalías presentes son en su mayoría peleas entre los individuos.

BOSS Dataset El conjunto de datos BOSS⁷ [10] es una colección de 19 escenas grabadas por las cámaras de seguridad de un tren, en las que varios grupos de gente (que abarca desde personas aisladas hasta grupos de más de 10 individuos) interaccionan de diversas formas, tanto normal como anormalmente. Para cada escena se tienen grabaciones desde distintos ángulos, gracias a distintas cámaras. Peleas, gente cayendo al suelo o situaciones de pánico son ejemplos de anomalías dentro de este conjunto.

UT Interactions Dataset El conjunto de datos UT Interactions⁸ [14] consiste en una colección de 20 vídeos de aproximadamente un minuto cada uno, los cuales muestran distintos tipos de interacciones entre dos personas: saludos, abrazos, empujones, patadas y puñetazos. Todos los vídeos presentan varias interacciones, así como individuos distractores (que no están implicados en la interacción). Las anotaciones están compuestas por rectángulos para señalar la posición de la anomalía junto con marcas temporales.

UCF-Crime Dataset El conjunto de datos UCF-Crime⁹ [3] es una colección de 1900 vídeos, 950 de ellos compuestos por eventos normales y 950 por eventos anormales. Los eventos anormales están además etiquetados en 13 clases distintas, por lo que el conjunto de datos puede utilizarse tanto para detección de anomalías como para clasificación de comportamientos. Este conjunto de datos es especialmente relevante debido a su gran tamaño, ya que está compuesto por más de 13 millones de fotogramas y una duración de más de 128 horas. No obstante, no ha sido ampliamente utilizado por el momento debido a su novedad, ya que fue publicado en 2018.

Peliculas Dataset y Hockey Fight Dataset Los conjuntos de datos Peliculas [15] y Hockey Fight [16] son dos colecciones de vídeos, de 200 y 1000

⁷BOSS está disponible en <http://velastin.dynu.com/videodatasets/BOSSdata/index.html>

⁸UT Interactions se puede descargar en https://cvrc.ece.utexas.edu/SDHA2010/Human_Interaction.html.

⁹La página del proyecto es <https://webpages.uncc.edu/cchen62/dataset.html>.

secuencias respectivamente, todos ellos de muy corta duración, alrededor de 5 segundos. Para cada conjunto de datos se tiene la mitad de escenas con comportamientos normales y la otra mitad con escenas de violencia (en el primer caso, utilizando vídeos extraídos de películas y en el segundo de partidos de hockey sobre hielo). Estos conjuntos de datos son ampliamente utilizados en sistemas de detección de violencia, que se considera una subtarea dentro de la detección de comportamientos anómalos.

2.3. Métricas de evaluación utilizadas en la detección de anomalías

El problema de detección de anomalías en multitudes puede ser visto como un problema de clasificación binaria. Para cada fotograma en el vídeo, se debe generar una etiqueta que indica si hay una anomalía presente en el mismo. Además, es un problema fuertemente desbalanceado, ya que la cantidad de fotogramas anómalos es muy inferior a la cantidad de fotogramas normales. En este contexto, suelen utilizarse métricas clásicas para la evaluación de clasificadores binarios para problemas desbalanceados. Además, dado que las predicciones pueden generarse en términos de fotograma o en términos de regiones dentro de cada fotograma, hay que realizar una distinción entre la evaluación a nivel de fotograma o la evaluación a nivel de píxel. En esta sección se estudia cómo pueden aplicarse ambos tipos de métricas al contexto de la detección de anomalías en multitudes.

2.3.1. Métricas clásicas

Las siguientes métricas se han encontrado recurrentemente en la revisión de la literatura llevada a cabo:

- Exactitud: Es la métrica más utilizada para los problemas de clasificación binaria. Se calcula como el número de predicciones correctas entre el número total de predicciones hechas. El principal problema de esta métrica es que aporta resultados engañosos cuando el conjunto de datos está desbalanceado, pero a pesar de este problema se utiliza ampliamente.
- Matriz de confusión: Dado un problema de clasificación con n clases distintas, la matriz de confusión es una matriz de tamaño $n \times n$, en la que el valor a_{ij} es un entero que representa el número de ejemplos de

la clase j que se han clasificado dentro de la clase i . Para un problema binario como el que nos ocupa, tenemos una matriz 2×2 . En relación con esta matriz, se definen una serie de métricas derivadas. En el contexto de la detección de anomalías en multitud, suelen calcularse las siguientes métricas derivadas:

- TPR: Ratio de verdaderos positivos, es la fracción de ejemplos positivos correctamente clasificados
- FPR: Ratio de falsos positivos, es la fracción de ejemplos negativos incorrectamente clasificados
- Precisión: Fracción de elementos bien clasificados de entre los clasificados como clase positiva
- F-score: Se define como la media armónica entre el TPR y la precisión. Es una medida de la calidad del modelo respecto de una determinada clase.
- Curva ROC y AUC: La curva ROC es una gráfica en la que se muestra la capacidad discriminativa de un modelo respecto a varios valores de discriminación. La métrica AUC es el área bajo la curva ROC del modelo. En problemas de clasificación binaria, se suele considerar que dados dos modelos, aquel con mayor AUC muestra un mejor comportamiento general.
- Tasa de error igual (Equal Error Rate, EER): Esta métrica mide el punto en el cual el ratio de falsa alarma y el ratio de fallo se igualan. Se puede calcular directamente a partir de la curva ROC. En la figura 2.3 se pueden observar tanto una curva ROC como el cálculo del EER.

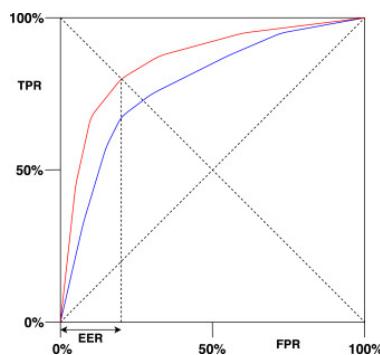


Figura 2.3: Ejemplo de curvas ROC y cálculo del EER. Se espera que el modelo con la curva roja tenga sea más potente, ya que su AUC es mayor. La intersección entre la curva ROC y la línea diagonal marca dónde el FPR y el TPR son iguales, punto que se define como el EER.

2.3.2. Métricas a nivel de fotograma y a nivel de píxel

Dado que la detección de anomalías en vídeo está muy relacionada con la detección de objetos en imágenes, algunos criterios de evaluación de anomalías son similares a los utilizados en la detección de objetos. En particular, se suelen utilizar dos tipos de métricas, las métricas a nivel de fotograma y las métricas a nivel de píxel.

Cuando se trabaja a nivel de fotograma, cada fotograma completo del vídeo se marca como normal o anómalo, sin tener en cuenta la posición que ocupa la anomalía en la escena. Cuando la evaluación se realiza a nivel de píxel, por el contrario, la anomalía se marca con una máscara sobre el fotograma, que indica la posición exacta donde se produce la anomalía. En este caso, la salida de los algoritmos debe ser una máscara comparable a la anotación (puede ser una matriz binaria con el tamaño de la imagen, las coordenadas de un rectángulo que encierre la anomalía, etc), y se considera que la anomalía se ha identificado correctamente si la máscara predicha coincide con la anotación original. El criterio usual para considerar esta asociación es lo que se conoce como métrica *Intersection over Union (IoU)*. Dado un rectángulo de anotación GT y un rectángulo de predicción PB, la IoU entre GT y PB se define como:

$$IoU(GT, PB) = \frac{|GT \cap PB|}{|GT \cup PB|}$$

Una anomalía se considera detectada si la métrica anterior entre la predicción y la anotación correspondiente es superior a un valor definido de antemano, usualmente $IoU(GT, PB) \geq 0.5$. Una vez se ha establecido la correspondencia entre las predicciones y las anotaciones, se realizan los siguientes cálculos: las predicciones correctamente asociadas a una anotación se consideran verdaderos positivos, las predicciones que no tienen una anotación asociada se consideran falsos positivos, y las anotaciones que no tienen una predicción son falsos negativos. Adicionalmente, si las anotaciones tienen una etiqueta de comportamiento normal o anómalo, aquellas anotaciones con etiqueta de comportamiento normal que no han recibido ninguna predicción son consideradas verdaderos negativos.

2.4. Revisión de la literatura

En esta sección se va a desarrollar una revisión bibliográfica de los trabajos que resuelven el problema de la detección de anomalías empleando redes

neuronales profundas en algún paso de la solución. Dividiremos la sección en función del tipo de anomalía que resuelvan, y tras esa división, agruparemos los trabajos en función del tipo de modelo empleado. En la sección 2.4.1 se muestran los trabajos que afrontan el problema de la detección de anomalías de movimiento y de apariencia. Se han combinado estos dos subproblemas dado que usualmente se resuelven conjuntamente. En particular, UCSD Pedestrians Dataset, el conjunto de datos más utilizado para la detección de movimiento anómalo, marca estos dos tipos de anomalía indistintamente en sus anotaciones, por lo que la mayoría de trabajos resuelven ambos problemas simultáneamente. En la sección 2.4.2 se presentan los trabajos que resuelven la detección de acciones anómalas, y en la sección 2.4.3 se expone el trabajo que se centra en detectar posiciones anómalas.

2.4.1. Detección de movimiento y apariencia anómala

En esta sección se revisan aquellos trabajos que utilizan modelos basados en aprendizaje profundo (Deep Learning, DL por sus siglas en inglés) para la detección de anomalías de movimiento y apariencia, agrupando dichos trabajos en función del tipo de modelo que utilizan para resolver la tarea.

Extracción de características con DL y SVM monoclase

Una aproximación común a este problema consiste en la utilización de modelos profundos para la extracción de características, y el uso de estas características en una etapa posterior de entrenamiento de un modelo de SVM monoclase. Las SVM monoclase [17] son una clase particular de SVM cuyo objetivo es encontrar la frontera de decisión más pequeña tal que los ejemplos normales con los que se ha entrenado queden dentro de dicha frontera. De esta forma, al clasificar nuevos ejemplos, aquellas características que quedan fuera de la región normal aprendida son clasificadas como anomalías.

En [18] se propone un modelo basado en tres *Denoising AutoEncoders* (estructuras autocodificadoras para eliminación de ruido) [19], que aprenden a reconstruir los fotogramas originales del vídeo (lo que extrae características de apariencia), el flujo óptico de los fotogramas [20] (lo que extrae características de movimiento) y la concatenación de ambos (que aprende características conjuntas). Una vez están entrenados los tres *AutoEncoders*, se utiliza la representación latente de cada modelo (es la capa en la que se obtiene la representación más reducida de cada fotograma) para entrenar tres SVMs monoclase. Finalmente, un modelo combinado por voto de las tres máquinas de soporte vectorial decide la clasificación final. De forma similar, Gutoski

et al. [21] Utiliza un *AutoEncoder* convolucional para reconstruir el fotograma de entrada, su flujo óptico y los ejes detectados con el detector de ejes de Canny de la imagen [22]. En una segunda etapa, se calculan los errores de reconstrucción de cada canal por separado, y se entrena una SVM sobre dichos 3 valores. Se realizan dos experimentos en este artículo. En el primero, se marcan regiones anómalas, lo que permite una evaluación a nivel de píxel. En el segundo, se marcan fotogramas enteros, utilizando el valor máximo encontrado en la imagen.

Los autores de [23] proponen un modelo combinado utilizando dos *Auto-encoders* de eliminación de ruido. Uno de ellos trabaja con el vídeo original, mientras que el otro recibe el primer plano de la escena, calculado utilizando el descriptor de Kanade-Lucas-Tomasi [24]. Para ambas entradas, se calcula el mapa de movimiento de la imagen y se introduce en el modelo. Una vez se han extraído las características en el espacio latente, se lleva a cabo una reducción de dimensionalidad con una red de creencia profunda (DBN) [25], y la representación reducida se introduce en la SVM para cada modelo. La puntuación de anomalía para cada fotograma se calcula finalmente como una combinación lineal de las salidas de ambos modelos. Se puede observar un esquema de este modelo en la figura 2.4.

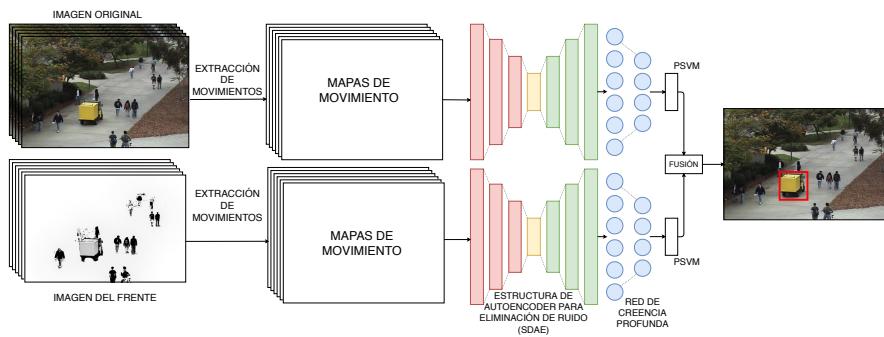


Figura 2.4: Ejemplo de SVM monoclasa para detección de anomalías. Dos estructuras autocodificadoras reciben los mapas de movimiento de las imágenes originales y sus correspondientes secuencias sin fondo, y tratan de reconstruir la información original. Se reduce la dimensionalidad de la representación latente con una red de creencia profunda y se utiliza una SVM monoclasa para aprender el patrón usual.

Fang et al. [26] utilizan dos características clásicas, los mapas de atención (saliency maps) [27] y los histogramas de flujo de movimiento multiescala

para entrenar una red neuronal profunda denominada PCANet [28]. Dicha red neuronal trata de aprender a hacer un análisis de componentes principales en cascada, lo que produce una reducción de dimensionalidad de la entrada. Tras la reducción de dimensionalidad, una SVM monoclasa aprende el patrón normal de características. En [29], se utiliza un modelo VGG-f [30] preentrenado como extractor de características, y se entrena una SVM sobre las características visuales extraídas. El uso de una red relativamente poco profunda hace que el modelo consiga funcionar a unos 20 fotogramas por segundo, lo que permite su uso en aplicaciones en tiempo real, consiguiendo unos resultados relativamente similares a los obtenidos con redes más profundas. De forma similar, Sun et al. [31] diseñan una capa especial de SVM monoclasa, la cual colocan al final de una red neuronal convolucional, de forma que pueden entrenar el modelo completo de una sola vez.

En [32] se propone un modelo de unión de tres modelos simples. Se entrena tres modelos basados en redes convolucionales (concretamente, un modelo VGG, un modelo AlexNet y un modelo GoogLeNet), y sus salidas se concatenan para formar vectores de características. Tras esto, los vectores de características se utilizan para entrenar distintos modelos de SVM monoclasa, cuya salida se combina para establecer la clasificación final.

Huang et al. [33] proponen también un modelo de combinación de características, extraídas utilizando máquinas de Boltzmann convolucionales [34]. Se entrena tres modelos distintos, utilizando como entrada regiones de la imagen original, regiones extraídas de la transformada por convolución gaussiana, y regiones extraídas del flujo óptico del vídeo. Las características extraídas sobre estas tres entradas se concatenan y se introducen en una SVM monoclasa que aprende los patrones de movimiento comunes.

Un ejemplo interesante puede encontrarse en [35]. Los autores proponen un modelo entrenado en dos fases. En una primera etapa, se entrena una versión modificada del detector Fast R-CNN [36] para que realice aprendizaje multitarea. Dicho entrenamiento se lleva a cabo de forma supervisada sobre conjuntos de gran tamaño, de forma que el modelo es capaz de extraer información semántica de los objetos que hay en la escena. En una segunda etapa, con el extracto semántico genérico ya entrenado, un detector de anomalías aprende el patrón normal específico para cada dataset en el que se quiera aplicar el método, y devuelve una puntuación de anomalía en tiempo de inferencia. En el artículo se ofrecen resultados con distintos detectores de anomalías, obteniéndose los mejores resultados con las SVMs monoclasa. La

ventaja de esta aproximación es su carácter genérico, ya que el extractor de características puede usarse en muchos contextos distintos. Esto hace que este método pueda utilizarse también para la detección de acciones anómalas. No obstante, se ha incluido en esta sección porque los autores muestran sus resultados utilizando el UCSD Peds1.

Extracción de características con DL y modelos gaussianos

Otro enfoque muy utilizado consiste en aprender una distribución de probabilidad sobre las características, usualmente utilizando distribuciones gaussianas, y considerando como anomalías aquellos ejemplos cuya probabilidad es baja respecto a la distribución aprendida. En este caso, de nuevo se extraen las características utilizando modelos de aprendizaje profundo.

En [37], se aprenden dos descriptores distintos. Por un lado, el vídeo se divide en parches 3D no solapados (regiones de la escena en varios fotogramas consecutivos), y se calculan descriptores globales y locales. Los descriptores locales se calculan a partir de una medida de similaridad entre el parche en cuestión y sus vecinos, mientras que el descriptor global es una representación latente aprendida con un modelo de *AutoEncoder*. En ambos casos, se aprende un modelo Gaussiano sobre los descriptores obtenidos, y en tiempo de inferencia se considera que un ejemplo es anómalo si ambos modelos marcan como anomalía a dicho parche. Los mismos autores ofrecen un refinamiento del modelo en [38]. En este caso, se utiliza el descriptor local como un primer filtro rápido para los parches sencillos. Cuando hay que clasificar un parche nuevo, si el modelo que ha aprendido sobre los descriptores locales devuelve como respuesta que el parche es normal, se termina la clasificación. Si se devuelve que el parche es anómalo, se computa el descriptor global y se clasifica con su modelo, que establece la clasificación final. De esta forma, se consigue un modelo con la potencia del descriptor global, que obtiene mejores resultados, pero con una reducción importante de cómputo, ya que el descriptor local es más sencillo de calcular.

Los mismos autores proponen también un modelo basado en cascada en [39]. En este caso, el primer discriminante funciona con el error de reconstrucción obtenido por un *AutoEncoder* poco profundo, que elimina rápidamente los parches simples (especialmente los parches compuestos por fondo, ya que el error de reconstrucción en este caso es muy bajo). A continuación, una red convolucional en tres dimensiones se entrena para extraer características de los parches no descartados. Con las características extraídas por la

red, se aprende el modelo Gaussiano, utilizando la distancia de Mahalanobis entre las características de los nuevos parches y los parches de entrenamiento.

Feng et al. [40] proponen un método basado en gradientes en tres dimensiones. Para cada vídeo a clasificar, se calculan los gradientes en vertical, horizontal, y dimensión temporal, y se entrena una red PCANet [28] sobre dichos descriptores. Sobre las características extraídas por la red, se entrena un modelo de mezcla de gaussianas profundo [41]. Dicho modelo es una adaptación de los modelos de mezcla de gaussianas de forma que pueden entrenarse como una red neuronal, utilizando optimización basada en el gradiente descendente. En tiempo de inferencia, si la salida del modelo gaussiano profundo está por debajo de un umbral, significa que la probabilidad de que la entrada ocurra es baja, y por tanto se marca el fotograma como anomalía.

Modelos basados en técnicas de reconstrucción

La idea que subyace en el funcionamiento de estos modelos consiste en entrenar un modelo capaz de reconstruir la imagen de entrada original a partir de su representación latente. Tras entrenar los modelos utilizando solamente imágenes normales, el resultado de aplicar estos modelos sobre imágenes anómalas produce reconstrucciones irregulares. Por tanto, calculando el error cometido en la reconstrucción se obtiene una medida de la irregularidad del fotograma de entrada.

Un ejemplo de esta técnica se puede encontrar en Ramchandran et al. [42]. Los autores entrena un *AutoEncoder* convolucional (CAE) con estructura recurrente basada en un modelo LSTM. Dicho modelo es un CAE con una configuración recurrente, de forma que puede trabajar directamente con secuencias de vídeo, y no exclusivamente con imágenes. En particular, el modelo LSTM-CAE (como lo denominan los autores), aprende a reconstruir fragmentos originales del vídeo a partir de las aristas detectadas en sus fotogramas utilizando el detector de Canny. Cuando un segmento anormal de vídeo es reconstruido en tiempo de inferencia, el error de reconstrucción cometido crece considerablemente, lo que permite definir un umbral a partir del cual se considera que un fragmento es anómalo.

Otro ejemplo de error de reconstrucción se encuentra en [43]. En este trabajo, se construye una CNN con estructura temporal y salida binaria, que traduce un vídeo de entrada en un conjunto de características binarias. Utiliza

lizando dichas características, se aprende un diccionario de códigos binarios, y cada fragmento de vídeo se representa como un histograma de dichos códigos. En tiempo de inferencia, la irregularidad de dicho histograma, medida como la cantidad de información que se pierde al representar el vídeo utilizando el diccionario, contiene información sobre el grado de anomalía del vídeo. Además, para calcular la región anómala con mayor precisión, se utiliza un sistema basado en el flujo óptico.

Modelos completos basados en aprendizaje profundo

Algunos autores han entrenado redes neuronales profundas cuya salida es directamente una puntuación de anomalía, en lugar de utilizar los modelos profundos como extractores de características sobre los que luego se entrena un modelo clásico. La principal ventaja de este enfoque es que los modelos pueden ser entrenados en un solo paso, sin necesidad de aplicar distintas etapas de entrenamiento.

Zhou et al. [44] definen un modelo espacio-temporal cuya salida es la probabilidad de que un determinado fragmento de vídeo contenga una anomalía. En el algoritmo, cada vídeo se divide en pequeños fragmentos, y se calcula el flujo óptico de los mismos. Si el parche tiene un objeto en movimiento, lo cual se mide en función de la magnitud del flujo, se considera que el fragmento es relevante y se procesa por la red neuronal. Si no se detecta movimiento con el flujo óptico, el fragmento se marca directamente como normal, para evitar procesamiento innecesario y reducir el tiempo de cómputo. De nuevo, este modelo se emplea también para la detección de acciones anómalas, no sólo de movimiento, pero al estar la mayoría del trabajo centrado en las anomalías de movimiento hemos decidido incluirlo en esta sección.

En [45] se entrena dos redes generativas adversarias (GANs [46]). En el primero de los modelos, el generador tiene que calcular el flujo óptico a partir de las imágenes originales, mientras que en el segundo se tiene que realizar la tarea inversa, es decir, calcular las imágenes originales a partir del flujo óptico. En ambos casos, las GANs se entrena exclusivamente sobre fotogramas normales. En tiempo de inferencia, sólo se utilizan los discriminantes, y dado que no se han entrenado sobre fotogramas anómalos, el modelo tiende a marcar los mismos como fotogramas falsos. La puntuación de anomalía se consigue finalmente sumando los valores de ambas redes.

Otros enfoques

Algunos de los modelos estudiados siguen enfoques completamente distintos a los comentados anteriormente. En [47], se propone un modelo en cascada. En una primera etapa, una CNN poco profunda descarta los parches de vídeo normales (usualmente los compuestos por fondo). Si dicha CNN no es capaz de marcar el fragmento como normal, se procesa el fragmento con una CNN más profunda. Las características visuales extraídas por la CNN se utilizan para estudiar el modelo de movimiento de los sujetos de la imagen utilizando un filtro de Kalman flexible. Dicho modelo es una modificación del filtro de Kalman clásico, en el que se mide cuánto se desvía el sujeto del modelo de movimiento normal. Dicha desviación se utiliza como medida de la anomalía.

La propuesta de Hu et al. tiene un enfoque interesante. En su trabajo proponen un nuevo modelo profundo, llamado D-IncSFA, cuyo objetivo es llevar a cabo una reducción de dimensionalidad utilizando una técnica llamada análisis de características lentas (*Slow Feature Analysis*, SFA [48]). Dicha técnica de reducción de dimensionalidad trabaja sobre series temporales, y trata de aislar aquellas características con cambio más lento que definen una señal. Dado que el cálculo de SFA es computacionalmente muy costoso, la red aprende a calcular una aproximación de dicha transformada. Tras ser entrenada sobre vídeos normales, la puntuación de anomalía se define como el cuadrado de las derivadas de la señal de salida. Dado que el modelo SFA es entrenado sólo sobre vídeos normales, al aplicarlo sobre vídeos anómalos la salida crece significativamente, lo que indica una anomalía. Se proponen dos soluciones distintas a partir de este modelo. Por un lado, para la detección de anomalías a nivel de fotograma, se toma la salida de la última capa y se calcula el máximo. Para la detección a nivel de píxel, se combina la salida de cinco capas distintas de la red, utilizándolas como mapas de características a distintas escalas.

2.4.2. Detección de acciones anómalas

En esta sección se revisan los modelos especializados en la detección de acciones anómalas. Estos trabajos no se han dividido por tipos de modelos ya que el número de trabajos dentro de este apartado es significativamente menor que en el caso anterior.

Siguiendo sus trabajos anterior en detección de movimiento anómalos, Sabokrou et al. [49] proponen un clasificador en cascada. Utilizando una red

completamente convolucional basada en AlexNet y preentrenada, se extraen parches de características multiescala para cada fotograma. A partir de esta información, se entrena dos clasificadores gaussianos. Un primer clasificador utiliza exclusivamente la información de las primeras k capas más superficiales de la red, y se aprende un clasificador simple. Si la predicción de dicho clasificador es segura (dos valores marcan si el comportamiento es normal o anormal con certeza), la clasificación se considera terminada. Si la predicción no es clara (se obtiene un valor en el intervalo central), se calculan características con capas más profundas para refinar la decisión, a costa de un coste computacional mayor.

Los errores de reconstrucción también se han utilizado para la detección de acciones anómalas. Un ejemplo de este tipo de modelos se encuentra en [50]. En este trabajo se entrena dos *AutoEncoders* para reducción de ruido capaces de extraer información visual y de movimiento de las trayectorias detectadas en el vídeo. Tras la extracción de características por parte de los *AutoEncoders*, se calcula una representación por bolsa de palabras, y el error de reconstrucción de las características originales a partir de su representación en la bolsa de palabras se considera una puntuación de anomalía.

En [3] se propone un modelo completamente basado en aprendizaje profundo para dar solución al problema. En su trabajo, la detección de anomalías se describe como un aprendizaje débilmente supervisado, desde el punto de vista del aprendizaje multiinstancia. Dado que en su trabajo utilizan un conjunto de datos etiquetado a nivel de vídeo, en lugar de a nivel de fotograma (esto es, los vídeos etiquetados como anómalos no marcan en qué fragmento del vídeo se sitúa la anomalía), el vídeo se divide en pequeños fragmentos que forman una bolsa de ejemplos. Los vídeos en los que se sabe que existe una anomalía forman bolsas positivas, y los vídeos normales forman bolsas negativas. Sobre los fragmentos de las bolsas se utiliza un extractor de características del tipo 3D CNN, y sobre dichas características se entrena una red completamente conectada, con una función de coste adaptada al aprendizaje multiinstancia. Nos centraremos en este modelo en capítulos posteriores, ya que nos servirá de base a la hora de diseñar nuestro modelo.

Los autores de [51] resuelven el problema de la detección de acciones anómalas utilizando un enfoque híbrido entre la detección de anomalías y la clasificación de eventos. Una red convolucional se entrena para distinguir entre 6 tipos distintos de eventos anómalos, dados un conjunto de fotografías de una secuencia de vídeo anómala. En el trabajo se llevan a cabo dos

experimentos distintos. En primer lugar, la salida de la red es una etiqueta binaria, la cual indica si en el vídeo se encuentra presente o no una anomalía. En el segundo experimento, la categoría concreta del comportamiento anómalo, si está presente, se especifica también como salida.

Las escenas de violencia han sido ampliamente estudiadas en el contexto de la detección de acciones anómalas. Uno de los primeros ejemplos del uso de aprendizaje profundo para la detección de violencia se puede encontrar en [52]. En este trabajo, se calculan cuatro mapas de características clásicos, todos ellos derivados del flujo óptico de la imagen. Estos mapas de características codifican información sobre orientación, magnitud y velocidad de dicho flujo. Esta información se procesa con un modelo AlexNet preentrenado que extrae características. A continuación, se lleva a cabo una selección de características utilizando el algoritmo Relief-F [53]. Con la representación reducida, se prueban dos detectores de anomalías, una SVM monoclasa y un modelo basado en k-NN. Ese mismo año, Sudhakaran et al. [54] proponen un modelo de entrenamiento en una fase compuesta por una red completamente convolucional seguida de una red recurrente LSTM para aprendizaje de los patrones de movimiento. Finalmente, un conjunto de capas completamente conectadas realizan la clasificación final. La particularidad de este modelo es que afronta el problema desde el punto de vista del aprendizaje supervisado, pero hemos decidido incluirlo aquí para dar una visión más completa de la detección de violencia. En la figura 2.5 puede observarse un esquema del funcionamiento de este modelo.

Una solución interesante se propone en el trabajo de Marsden et al. [55]. Dicho modelo es una red CNN multitarea que resuelve conjuntamente el problema de la detección de violencia, el conteo de individuos, y la estimación de densidad en multitudes. Las salidas de dicha red son una etiqueta binaria indicando la presencia o no de violencia en la imagen, una neurona de regresión que devuelve el número de individuos, y un mapa de calor que indica la densidad de personas. Además, en este trabajo se propone un conjunto de 100 imágenes completamente anotadas que se han usado para el entrenamiento del modelo. El principal inconveniente de este método es que funciona exclusivamente con imágenes, y no está preparado para su uso con secuencias de vídeo.

En [56] se propone un modelo basado en redes convolucionales 3D. Según los autores, la principal contribución de su trabajo es el método de selección aleatorio de fotogramas. Los fotogramas relevantes del vídeo se seleccionan

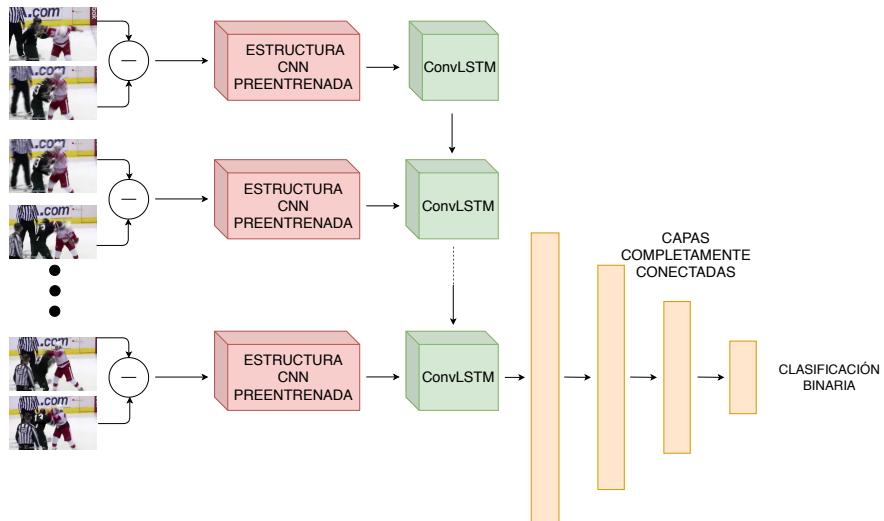


Figura 2.5: Ejemplo de arquitectura para detección de violencia. El modelo de la imagen utiliza una arquitectura llamada CNN-LSTM. Consiste en una red convolucional cuya salida es utilizada por una LSTM convolucional que aprende patrones temporales. Tras la extracción de patrones temporales, un grupo de capas densas produce la clasificación final. Imagen adaptada de [54].

utilizando un modelo clásico de agrupamiento. Dados dos fotogramas relevantes consecutivos, se eligen 16 fotogramas aleatorios entre ellos, y dicha información se introduce en la red convolucional para extraer características espacio-temporales. Estas características se introducen después en un detector de anomalías, el cual no se especifica en el texto del artículo. En [57], se propone un modelo LSTM bidireccional, modificado para ser utilizado en un entorno *Big Data*. Los vídeos son procesados dentro del ecosistema Spark, y una representación basada en el histograma de gradientes se calcula para cada fotograma. Esta representación se introduce después en el modelo recurrente, el cual aprende los patrones temporales de los eventos violentos. De nuevo, se resuelve el problema desde el punto de vista del aprendizaje supervisado.

Finalmente, en [58] se realiza un estudio comparativo entre distintos modelos basados en aprendizaje profundo. Los autores muestran resultados utilizando modelos basados en redes convolucionales y recurrentes, entrenadas con distintas políticas, y dando una comparación numérica entre ellos. El conjunto de datos utilizado está compuesto por varios vídeos extraídos de YouTube, donde se presenta una multitud bastante densa con diversos epi-

sodios de violencia. En este trabajo se concluye que los modelos preentrenados son los que mejores resultados arrojan, debido al reducido tamaño del conjunto de datos. Los modelos entrenados desde cero suelen dar resultados peores, a pesar de ser más complejos, por la cantidad de datos escasa de la que se dispone.

2.4.3. Detección de posiciones anómalas

Como hemos dicho anteriormente, este es el tipo de anomalía más sencillo de detectar. Una anomalía producida por posición se da cuando un sujeto no identificado entra en una zona restringida. Normalmente, el flujo de funcionamiento de estos modelos consiste simplemente en una detección de peatones, seguida de una etapa de solapamiento entre los rectángulos detectados y las regiones del fondo restringidas.

Este flujo es utilizado en el trabajo de Cheng el al [59]. En este trabajo, se considera anomalía la incursión en un área restringida dentro de un puerto. Los vídeos están recogidos por una cámara aérea, y el modelo está compuesto por un detector simple (SSD [60]) refinado para la detección exclusiva de peatones y una zona ilegal definida en el fondo de la imagen. El sistema lanza una alarma cuando un trabajador del puerto penetra en la zona ilegal.

3

Características espacio-temporales para la detección de acciones anómalas

En este capítulo estudiaremos el uso de características espacio-temporales para la detección de acciones anómalas en vídeo. Concretamente, tomaremos como punto de partida el modelo propuesto en el trabajo *Real-World Anomaly Detection in Surveillance Videos* [3], y realizaremos una propuesta de mejora. Nuestro enfoque consiste en cambiar el extracto de características empleado originalmente, el cual se basa en el uso de convoluciones 3D [61], por uno basado en convoluciones clásicas 2D seguido de una etapa recurrente.

Nuestra hipótesis defiende que, aunque las convoluciones 3D están pensadas para extraer información de fragmentos de vídeo, no son capaces de capturar correctamente la dependencia temporal a largo plazo. La falta de capacidad en este sentido produce un empeoramiento global del modelo, el cual puede subsanarse con un extracto de características más potente, que haga uso tanto de capas convolucionales para extraer información espacial como de capas recurrentes para aprender la secuencia temporal.

En esta sección se detallará la experimentación llevada a cabo, describiendo el conjunto de datos utilizado, el modelo original, y las modificaciones propuestas.

3.1. Conjunto de datos empleado: UCF-Crime Dataset

El conjunto de datos con el que se ha trabajado fue propuesto por los propios autores del artículo que estamos estudiando¹. Dicho conjunto de datos está compuesto por secuencias de vídeo extraídas de cámaras de videovigilancia. Según puede consultarse en la página del proyecto, la base de datos está compuesta por 1900 vídeos de longitud variable, con una media de 7247 fotogramas. Estamos hablando, por tanto, de vídeos de una longitud acep-

¹Puede consultarse la página del proyecto en <https://www.crcv.ucf.edu/projects/real-world/>

table, de varios minutos de media. En total, se dispone de 128 horas de vídeo, y se incluyen 13 clases distintas de anomalías, así como vídeos que se consideran normales. Se consideran vídeos normales aquellos en los que no aparecen comportamientos anómalos. Algunas de las clases de anomalía que se incluyen en el conjunto son abusos, robos, allanamientos de morada, tiroteos, asaltos a comercios, peleas o explosiones.

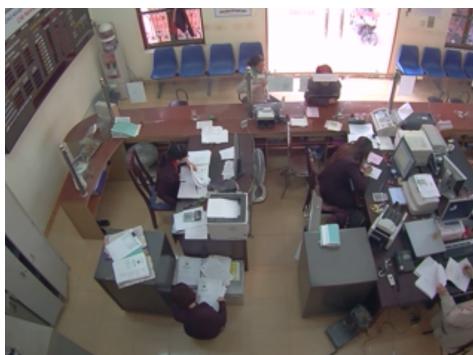
Una particularidad del conjunto de datos es la gran diversidad de entornos en los que se recogen los vídeos. En la figura 3.1 se pueden observar distintos fotogramas extraídos de vídeos normales. Como se puede apreciar, tenemos vídeos tomados por cámaras interiores y exteriores, tanto a la luz del día como por la noche, y con distintos ángulos de cámara. Esto limita la posibilidad de hacer asunciones sobre los datos, y obliga a construir un modelo genérico, capaz de funcionar en diversas situaciones.



(a) Interior a la altura de la vista



(b) Exterior de día



(c) Interior desde arriba



(d) Exterior de noche

Figura 3.1: Ejemplos de fotogramas normales del conjunto de datos UCF-Crime. Podemos observar la gran variabilidad de las escenas del conjunto.

Se proponen dos problemas distintos. Por un lado, se pide diseñar un modelo que sea capaz de marcar cuándo ocurre una anomalía (es decir, un problema binario, en el que se debe marcar para cada fotograma del conjunto de

datos si presenta o no anomalía), y por otro, para los vídeos etiquetados como anómalos, asignar correctamente la clase a la que pertenecen. En nuestro caso, nos centraremos en resolver el primero de los problemas. Los autores del modelo original dan mucha más importancia a la primera parte que a la segunda, y por tanto nosotros nos hemos centrado en su análisis, ignorando la segunda parte. Por tanto, para nosotros no habrá distinción entre las distintas clases de anomalías, por lo que a partir de este momento sóloaremos la distinción entre vídeos normales y anómalos. El conjunto de datos está dividido en subconjuntos de entrenamiento y test. Para el conjunto de entrenamiento se dispone de 800 vídeos normales y 810 vídeos anómalos. Para el conjunto de datos de test, se dispone de 290 vídeos (150 normales, 140 anómalos).

La principal particularidad del conjunto de datos es que el subconjunto de entrenamiento está débilmente etiquetado. Esto significa que, aunque el objetivo a abordar consista en localizar temporalmente cuándo ocurre una anomalía, las etiquetas de las que se dispone en el conjunto de entrenamiento sólo marcan que el vídeo es anómalo, no dónde ocurre dicha anomalía. Por tanto, en lugar de tener para cada vídeo un valor binario por fotograma indicando si hay una anomalía o no presente en dicho fotograma, lo que se tiene es una única etiqueta binaria para el vídeo completo. Como se puede observar en la figura 3.2, en los vídeos que presentan anomalía hay también una parte importante del vídeo correspondiente a fotogramas normales.

En el conjunto de test, sin embargo, el etiquetado es a nivel de fotograma, y por tanto tendremos que aprender a localizar temporalmente la anomalía en un fragmento de mayor longitud. Esto hace que haya que diseñar un sistema de aprendizaje específico, que permita aprender a nivel de fotograma cuando no tenemos dicha información disponible.

Sobre las etiquetas del conjunto de test, observamos que nos encontramos ante un problema fuertemente desbalanceado. En total, disponemos de 1027477 fotogramas etiquetados como fotogramas normales, mientras que sólo hay 84331 fotogramas anómalos. Resulta sorprendente el hecho de que, a pesar de tener los conjuntos prácticamente balanceados en términos de vídeos anómalos y normales, cuando realizamos el cálculo a nivel de fotograma, existe un desbalanceo muy importante. Esto se debe a que, en realidad, los vídeos que presentan una anomalía concentran la misma en unos pocos segundos, y la mayoría del tiempo de vídeo está compuesto por fotogramas normales.



(a) Fuego - Fotograma normal



(b) Fuego - Fotograma anómalo



(c) Explosión - Fotograma normal



(d) Explosión - Fotograma anómalo



(e) Accidente de tráfico - Fotograma normal



(f) Accidente de tráfico - Fotograma anómalo



(g) Robo - Fotograma normal



(h) Robo - Fotograma anómalo

Figura 3.2: Ejemplos de anomalías del conjunto de datos UCF-Crime. Como puede observarse en cada pareja de fotogramas, en los videos anómalos hay también fragmentos normales presentes.

A continuación describiremos el modelo original, así como la política de entrenamiento del mismo.

3.2. Modelo original

En esta sección se va a describir la arquitectura original empleada. Dicha arquitectura consiste en un extracto de características basado en redes neuronales convolucionales en tres dimensiones, seguido por una red completamente conectada que realiza la clasificación final. Además, la principal aportación del modelo es una función de pérdida que permite aprender el etiquetado a nivel de fotograma a pesar de entrenar con etiquetado a nivel de vídeo. Esta función resuelve la particularidad del etiquetado del conjunto de datos de entrenamiento de forma débil.

En las siguientes secciones describiremos en detalle cada una de las partes de la red, así como la función de pérdida propuesta.

3.2.1. Extractor de características: C3D

El extracto de características empleado en el modelo original es el modelo conocido como C3D [61]. La particularidad de este modelo es que en lugar de utilizar convoluciones en dos dimensiones, que son las que suelen usarse típicamente en el tratamiento de imágenes, se utilizan convoluciones en tres dimensiones. La diferencia radica en que la convolución en dos dimensiones sólamente desplaza el kernel a lo largo del ancho y el alto de la imagen. El núcleo de convolución es, por tanto, una matriz, y la salida de la convolución es en dos dimensiones.

Cuando se aplica una convolución 3D, se tiene en cuenta también la dimensión temporal. En este caso, los núcleos son tensores de rango 3, y la convolución no utiliza un único fotograma, si no que tiene en cuenta varios fotogramas consecutivos. De esta forma, no sólo se recoge información espacial (en las dimensiones alto y ancho), sino que también se recoge información temporal (dado que se involucra la dimensión tiempo). Así, la salida es en tres dimensiones, en lugar de en dos. En la imagen 3.3 puede observarse gráficamente la diferencia entre una convolución en dos dimensiones y una convolución en tres dimensiones.

El extractor propuesto consiste, por tanto, en una serie de convoluciones 3D que toman como entrada un vídeo y extraen características del mismo.

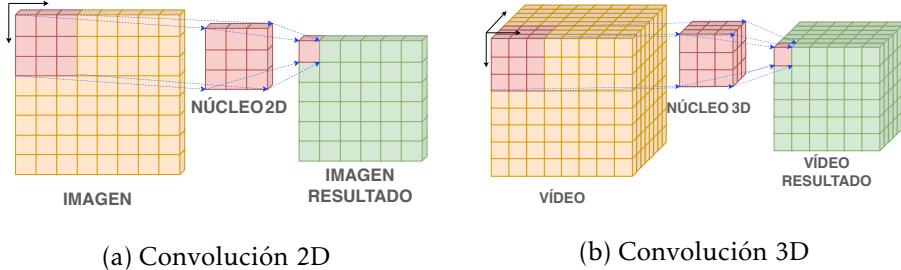


Figura 3.3: Diferencia entre la convolución en dos y en tres dimensiones. Como puede observarse en la imagen, en la convolución 2D el filtro es una matriz que se desplaza en dos dimensiones. En la convolución 3D, por el contrario, el filtro es 3D y se mueve también en la dimensión temporal.

En concreto, la arquitectura del modelo C3D es la siguiente:

- Convolución 3D, núcleo $3 \times 3 \times 3$, 64 filtros
- Pooling 3D, reducción de factor 2 en las dimensiones espaciales, sin reducción en la dimensión temporal
- Convolución 3D, núcleo $3 \times 3 \times 3$, 128 filtros
- Pooling 3D, reducción de factor 2 en las tres dimensiones
- Convolución 3D, núcleo $3 \times 3 \times 3$, 256 filtros
- Convolución 3D, núcleo $3 \times 3 \times 3$, 256 filtros
- Pooling 3D, reducción de factor 2 en las tres dimensiones
- Convolución 3D, núcleo $3 \times 3 \times 3$, 512 filtros
- Convolución 3D, núcleo $3 \times 3 \times 3$, 512 filtros
- Pooling 3D, reducción de factor 2 en las tres dimensiones
- Convolución 3D, núcleo $3 \times 3 \times 3$, 512 filtros
- Convolución 3D, núcleo $3 \times 3 \times 3$, 512 filtros
- Pooling 3D, reducción de factor 2 en las tres dimensiones
- Capa completamente conectada, 4096 neuronas

- Capa completamente conectada, 4096 neuronas
- Capa de activación *softmax*

Según los autores del modelo, la salida de la primera capa completamente conectada es la que debe utilizarse como representación del fragmento de vídeo. La entrada del modelo es un vídeo de 16 fotogramas de tamaño 112×112 , y devuelve como resultado un descriptor de 4096 elementos de dicho fragmento de vídeo.

Preentrenamiento del modelo: Sports-1M Dataset

El extracto de características empleado en el trabajo se entrena previamente en un conjunto de datos cuyo objetivo es la clasificación de comportamientos. Este conjunto de datos de gran tamaño suele utilizarse como punto de partida para el entrenamiento de modelos dedicados la extracción de características en vídeo, de forma similar al uso de ImageNet para imágenes.

Dado que las redes neuronales utilizadas en trabajos de esta envergadura suelen necesitar de conjuntos de datos de gran tamaño para ser entrenadas, los conjuntos que se utilizan normalmente no son suficientemente grandes para que el aprendizaje sea suficientemente rico. Por este motivo, suele emplearse una etapa de entrenamiento previa sobre un conjunto de datos de gran tamaño, con el que se aprenden características genéricas, y después se afina el modelo en el conjunto de datos con el que se trabaja. Esta técnica es la que se conoce como transferencia de aprendizaje (*Transfer learning*).

En concreto, en el trabajo que estamos analizando se entrena el extracto de características en el conjunto de datos Sports-1M [62]. Dicho conjunto está formado por 1133158 vídeos, con un total de 487 clases distintas. El extracto de características se entrena colocando una capa densamente conectada al final de la arquitectura anterior, con tantas neuronas como clases tiene el conjunto de datos (487 en este caso), y entrenando la red completa para resolver la tarea de clasificación. Una vez entrenada la red para resolver dicho problema, debido a que los comportamientos presentes en el conjunto de datos son muy variados, las capas de la red son capaces de extraer información muy diversa de los vídeos de entrada. Para construir el extracto de características final, se elimina la última capa de clasificación y se mantienen los pesos aprendidos para el resto de capas. Esta última técnica es lo que se conoce como congelar la red.

Tras este entrenamiento, obtenemos un modelo capaz de resumir la información de un vídeo de 16 fotogramas en un vector de 4096 componentes. Se utilizará este modelo para resumir la información de cada vídeo del conjunto en fragmentos de 16 fotogramas. De esta forma, por cada vídeo, tendremos un número variable de descriptores extraídos. En la siguiente sección analizaremos cómo se convierten dichos descriptores en una representación de tamaño fijo del vídeo, y como puede utilizarse esa información para entrenar el clasificador final. Para ello, tendremos que definir una función de coste que nos permita aprender a nivel de fotograma (o de unos pocos de fotogramas), a pesar de tener etiquetas sólo a nivel de vídeo.

3.2.2. Aprendizaje multi-instancia

En esta sección se presenta la política de aprendizaje diseñada en el artículo para el entrenamiento del modelo. Como hemos comentado anteriormente, tenemos etiquetas a nivel de vídeo, pero no a nivel de fotograma, que es la tarea que realmente queremos resolver.

La propuesta de entrenamiento se basa en lo que se conoce como aprendizaje multi-instancia. En lugar de recibirse una etiqueta para cada elemento del conjunto de datos, se recibe una bolsa con elementos etiquetados de forma única. En el caso de la clasificación binaria, una bolsa estará etiquetada como negativa si todos los elementos que la conforman son elementos negativos. Por otro lado, una bolsa tendrá etiqueta positiva si al menos uno de sus elementos es positivo. Esto se traslada de forma muy sencilla a nuestro escenario. En nuestro caso, cada vídeo representará una única bolsa, etiquetada como negativa si procede de un vídeo normal (en el que no ocurren anomalías), y positiva procede de un vídeo anormal (sabemos que ocurre una anomalía en dicho vídeo, pero no dónde). Para conseguir una representación de tamaño fijo para cada vídeo, se dividen los mismos en 32 segmentos temporales, lo que hace que en cada bolsa haya 32 elementos. Para definir lo que ocupa cada segmento, se reparten de forma equiespaciada los fragmentos de 16 fotogramas procesados anteriormente, de forma que en cada segmento haya el mismo número de fragmentos, todos ellos consecutivos (o el reparto más equitativo posible, si la división no es exacta). La representación del segmento es la media de las representaciones de los fragmentos que lo forman. De esta forma, para cada vídeo se obtienen 32 elementos distintos, que conformarán la bolsa en cuestión.

Ahora, tenemos que definir la función de pérdida con la que entrenar

el modelo, dadas las bolsas que hemos obtenido. En un contexto normal, si tuviéramos ejemplos de vídeos anómalos \mathcal{V}_a y normales \mathcal{V}_n etiquetados completos, queríramos conseguir un modelo cuya salida fuese una puntuación de anomalía, tal que

$$f(\mathcal{V}_a) > f(\mathcal{V}_n).$$

No obstante, lo que tenemos son bolsas de ejemplos, en lugar de vídeos completos. En dichas bolsas sabemos que existen ejemplos positivos, pero no sabemos cuáles son. Dadas dos bolsas de ejemplos, una proveniente de un vídeo normal \mathcal{B}_n y una proveniente de un vídeo anómalo \mathcal{B}_a , querremos conseguir que la puntuación del segmento de puntuación más alta de la bolsa normal (aquel trozo de vídeo que siendo normal resulta difícilmente clasificable) sea más alto que el segmento de puntuación más alta de la bolsa anormal (aquel segmento de la bolsa anómala con una probabilidad mayor de ser anómalo). Es decir, nuestra intención será forzar que

$$\max_{i \in \mathcal{B}_a} f(\mathcal{V}_a^i) > \max_{i \in \mathcal{B}_n} f(\mathcal{V}_n^i).$$

Con esta formulación, queremos conseguir que los fragmentos que contienen realmente una anomalía tengan una puntuación mayor que aquellos que no la contienen, pero que son ejemplos difícilmente clasificables. Utilizando una idea similar a la función de pérdida de Hinge (Hinge loss), la cual se utiliza en el entrenamiento de las SVM para maximizar el margen de clasificación, se define la función de pérdida para el entrenamiento del modelo como

$$l(\mathcal{B}_a, \mathcal{B}_n) = \max(0, 1 - \max_{i \in \mathcal{B}_a} f(\mathcal{V}_a^i) + \max_{i \in \mathcal{B}_n} f(\mathcal{V}_n^i)).$$

Además, en la función anterior hemos ignorado la estructura temporal del vídeo de entrada. En un vídeo con anomalías, tenemos que la mayoría del tiempo nos encontramos con fotogramas normales. Por este motivo, queremos que la mayoría de las predicciones que hagamos sean cercanas a cero. Además, nos interesa evitar que el cambio en la puntuación de anomalías sea excesivamente abrupto, por lo que buscamos que el aumento de dicho valor entre un segmento y el siguiente no sea demasiado fuerte. Esto nos lleva a añadir dos términos de regularización a la función de coste, que se encarguen de controlar las dos cantidades que hemos mencionado. Con dicha modificación, la función de pérdida final se convierte en

$$l(\mathcal{B}_a, \mathcal{B}_n) = \max(0, 1 - \max_{i \in \mathcal{B}_a} f(\mathcal{V}_a^i) + \max_{i \in \mathcal{B}_n} f(\mathcal{V}_n^i)) \\ + \lambda_1 \sum_{i=0}^{n-1} (f(\mathcal{V}_a^i) - f(\mathcal{V}_a^{i+1})) + \lambda_2 \sum_{i=0}^n f(\mathcal{V}_a^i).$$

El término que acompaña al parámetro λ_1 hace referencia al término de regularización temporal, y el término que acompaña al parámetro λ_2 hace referencia a la regularización dispersa. Los valores λ_1 y λ_2 pueden ajustarse para dar más o menos importancia a los términos de regularización.

Una vez hemos visto la política de entrenamiento del modelo clasificador, pasamos a dar la descripción completa del modelo original.

3.2.3. Modelo original completo

Una vez hemos visto el extractor de características y la política de entrenamiento, pasamos a ver cómo se entrena el modelo original completo.

En la figura 3.4 se puede observar la estructura completa del modelo original. Dados dos vídeos del conjunto de datos, uno normal y uno anómalo, se extraen sus 32 segmentos temporales y se conforman las bolsas de ejemplos. Los 32 segmentos se procesan con el extractor de características preentrenado, y se extraen los descriptores resumen para cada segmento. Dichos descriptores se clasifican utilizando una red completamente conectada, la cual tiene como única salida una neurona. Dicha neurona aprende a devolver una puntuación de anomalía para cada segmento. Cuando se han calculado las puntuaciones de anomalía para los vídeos positivo y negativo, se calcula el valor de la función de pérdida y se actualizan los pesos de las capas densas utilizando un método de descenso del gradiente. El extractor de características está preentrenado y congelado, por lo que en tiempo de entrenamiento del modelo completo no se actualizan sus pesos.

Para cada etapa de entrenamiento, se seleccionan aleatoriamente 60 vídeos del conjunto de forma balanceada (30 normales y 30 anómalos), y se calcula el gradiente de la función de pérdida en función de la clasificación de los 60 vídeos.

Una vez que se tiene el modelo entrenado con el sistema anterior, en tiempo de inferencia se divide el vídeo a clasificar de la misma forma que en

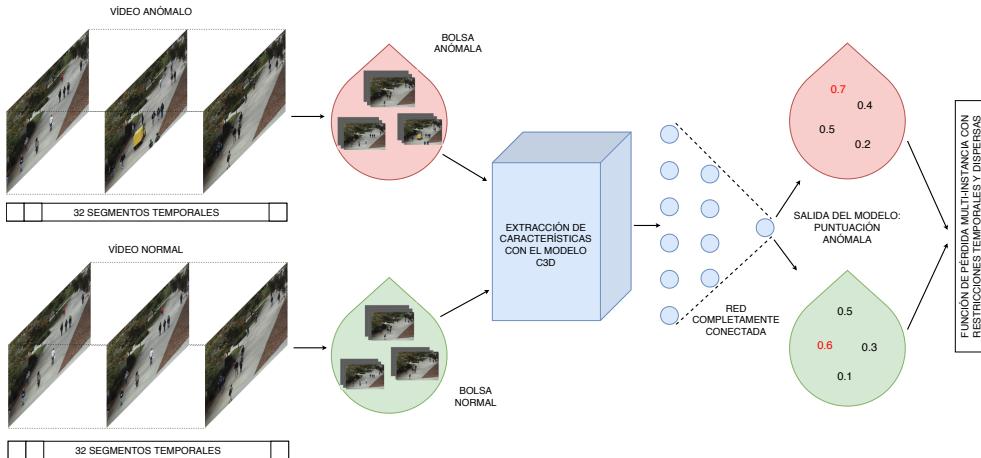


Figura 3.4: Arquitectura completa del modelo original

entrenamiento, y se calcula la puntuación de anomalía para cada segmento. Para obtener una predicción para cada fotograma, se interpola la predicción de los segmentos.

Una vez hemos visto el modelo original al completo, pasamos a describir la modificación propuesta.

3.3. Propuesta de mejora del modelo

La propuesta de mejora del modelo consiste en una modificación del extractor de características por un sistema que capture correctamente la información temporal contenida en los fragmentos de vídeo.

La principal crítica que se puede hacer al modelo anterior radica en que el extractor de características no es el idóneo. A pesar de que las convoluciones en tres dimensiones son modelos pensados para la extracción de características en vídeo, su capacidad de extraer información temporal contenida en un número elevado de fotogramas consecutivos es limitada, ya que el campo receptivo de la red viene limitado por el tamaño de los núcleos utilizados. En el caso del modelo anterior, trabajamos con núcleos de tamaño 3, por lo que extraeremos información que esté expresada en tres fotogramas consecutivos, no a mayor distancia. Gracias la operación de *Pooling*, en capas más avanzadas de la red se extrae información de fotogramas que sí se encuentran a mayor distancia, pero en nuestra opinión la dependencia temporal está infrarepresentada en estos modelos.

Nuestra propuesta, por tanto, trata de solucionar esta problemática proponiendo un extractor de características espacio-temporales de mayor calidad, el cual se expone a continuación:

3.3.1. Extractor de características: Xception-LSTM

Como hemos comentado anteriormente, propuesta se basa en modificar el extractor de características del modelo anterior, de tal forma que se utilice una arquitectura con capacidad de capturar correctamente la información temporal. Para ello, haremos uso de redes neuronales recurrentes. Este tipo de modelos se distinguen de las redes neuronales clásicas en el hecho de que conectan la salida de neuronas de una capa con entradas de neuronas dentro de la misma capa. De esta forma, la predicción que realizan no depende sólo de un único elemento (o un conjunto pequeño de elementos) del vector de entrada, sino también de la información que ha recibido la red en etapas anteriores. En la figura 3.5 se puede observar la distinción entre una red neuronal clásica y una red recurrente. En particular, utilizaremos redes convolucionales en dos dimensiones para extraer información de todos los fotogramas del vídeo, y crearemos una serie temporal con dichos descriptores. La serie temporal será utilizada como entrada para una red recurrente que aprenderá el patrón temporal.

Concretamente, utilizaremos una arquitectura LSTM [63]. Esta arquitectura fue propuesta para dar solución al problema del olvido que se detectó en las primeras redes neuronales recurrentes. Este problema consiste en que, aunque teóricamente las redes neuronales recurrentes deben ser capaces de aprender patrones temporales de larga duración, se observó que cuando la información estaba suficientemente separada la efectividad de estos modelos se deterioraba significativamente. Para mitigar este deterioro, lo que utiliza la red neuronal LSTM es un estado interno que se transmite entre celdas, además de la propia salida de la red. De esta forma, se separa en parte la salida de cada neurona y el estado interno de la red que se transmite a las neuronas siguientes. Se observó que esta separación resulta beneficiosa y mejora significativamente los resultados que obtiene la red. En la figura 3.6 puede observarse la estructura interna de una neurona de una LSTM. El funcionamiento básico de estas neuronas consiste en combinar la memoria de la red (representada por el estado interno y la salida previa) con la entrada actual por medio de tres módulos distintos, conocidos como puertas. En la imagen podemos reconocer las puertas por las conexiones que se realizan desde abajo

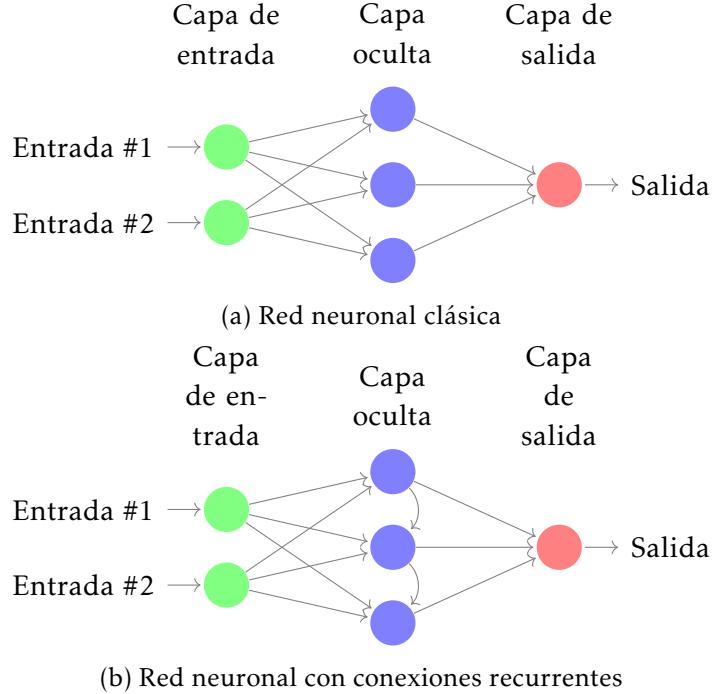


Figura 3.5: Diferencia entre una red neuronal clásica y una red neuronal recurrente. Como podemos observar, en la red recurrente tenemos conexiones entre las neuronas de la capa oculta, lo cual no ocurre en las redes neuronales clásicas.

hacia arriba. El funcionamiento de las puertas es el que sigue:

- **Puerta del olvido:** Es la primera de las puertas que encontramos en el esquema. Se encarga de decidir qué información se mantiene de la proveniente del estado interno anterior.
- **Puerta de entrada:** Segunda de las puertas del esquema. Con ella se regula la información de entrada que se incorpora al estado interno de la red. Con la información de la puerta del olvido y la puerta de entrada se calcula el nuevo estado interno de la red.
- **Puerta de salida:** Última puerta del esquema. Combina el nuevo estado interno con la salida y la entrada previas para definir la salida de la capa.

Una vez hemos introducido las redes neuronales recurrentes, comentaremos la arquitectura extractora completa que hemos propuesto.

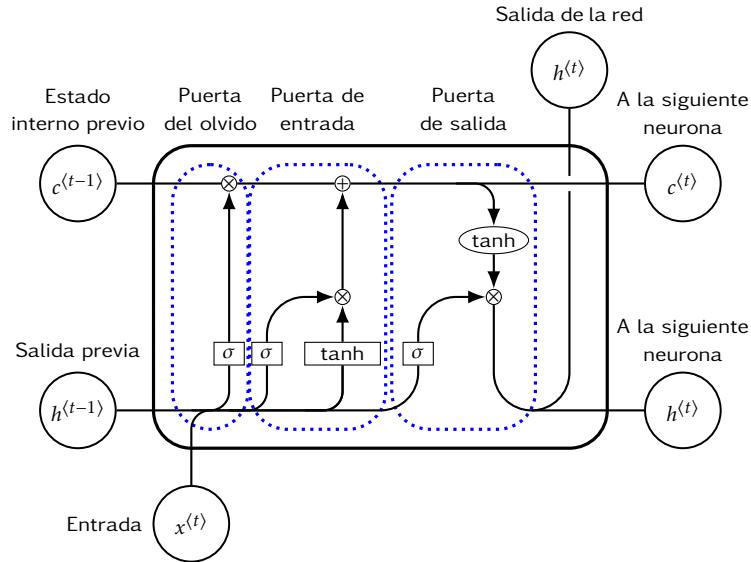


Figura 3.6: Estructura de una neurona LSTM. Cada neurona recibe, además de la entrada correspondiente, el estado interno y la salida previos de la capa.

Al igual que en el modelo original, trabajaremos con fragmentos de vídeo de 16 fotogramas. Introduciremos independientemente los fotogramas en un modelo Xception [64], preentrenado sobre el conjunto de datos ImageNet [65] (dicho modelo está disponible para descargar directamente en el framework Keras [66], que se ha utilizado para la implementación). Como descriptor de cada fotograma se utiliza la salida de la penúltima capa de la red, la cual es una capa de tipo *Global Average Pooling*. Dicha capa resume toda la información de un mapa de características haciendo la media de los valores de todos los píxeles que lo conforman. La salida que se obtiene es un vector de 2048 valores.

En consecuencia, la representación por fotogramas de un vídeo es una matriz de dimensiones 16×2048 . Dicha representación se introduce en una capa LSTM, que considerará la entrada como una serie temporal de 16 instantes de tiempo con 2048 características.

La salida de dicha capa será un descriptor que representará la información espacio-temporal contenida en el fragmento de 16 fotogramas de vídeo. Se han realizado experimentos con diferentes tamaños de capa. En concreto, se han propuesto descriptores de tamaño 512, 768 y 1024, obteniéndose mejores resultados cuanto mayor era el tamaño del descriptor.

Preentrenamiento del modelo: UCF-101

Dada la falta de recursos computacionales que hemos experimentado a la hora de entrenar los modelos, en lugar de haber preentrenado el extracto de características en el conjunto Sports-1M (en el que se preentrenó el extracto original), hemos preentrenado nuestro extracto en el conjunto de datos UCF-101 [67]. Dicho conjunto cuenta con 13320 videos repartidos en 101 clases distintas. La duración de los videos es relativamente corta, desde poco más de un segundo hasta algo más de un minuto en los videos más largos.

Como podemos observar, nos encontramos ante un conjunto de datos de un tamaño mucho menor que el utilizado por los autores del trabajo original. Esta diferencia de tamaño producirá que las características que aprendamos con nuestro extracto de características sean menos ricas y variadas que las que podríamos aprender con un conjunto mayor, lo cual puede traducirse en una pérdida de rendimiento del modelo. No obstante, estamos hablando de un conjunto de datos de tamaño aceptable para plantearnos el preentrenamiento del modelo.

En cuanto a la política de entrenamiento, añadiremos al extracto de características un bloque de capas densamente conectadas al final, que llevarán a cabo la clasificación de los videos. Para crear los ejemplos de entrenamiento, de los videos del conjunto de datos UCF-101 seleccionaremos 16 fotogramas equiespaciados dentro del video completo, y escalaremos los fotogramas para que tengan el tamaño de entrada requerido por la red Xception (la entrada tendrá un tamaño final de $16 \times 299 \times 299$). Los pesos de la red convolucional estarán congelados, para reducir la carga computacional del entrenamiento, ya que consideramos que las características aprendidas en ImageNet son de buena calidad para la clasificación de imágenes. Por tanto, sólo entrenaremos la capa recurrente y las capas completamente conectadas.

Una vez se ha llevado a cabo el entrenamiento completo, eliminamos las capas densamente conectadas, y utilizamos la salida de la capa recurrente como descriptor del fragmento de video. Tenemos, por tanto, un descriptor de 512, 768 o 1024 elementos para cada fragmento de 16 fotogramas. En la figura 3.7 podemos observar el extracto de características completo, preparado para el preentrenamiento.

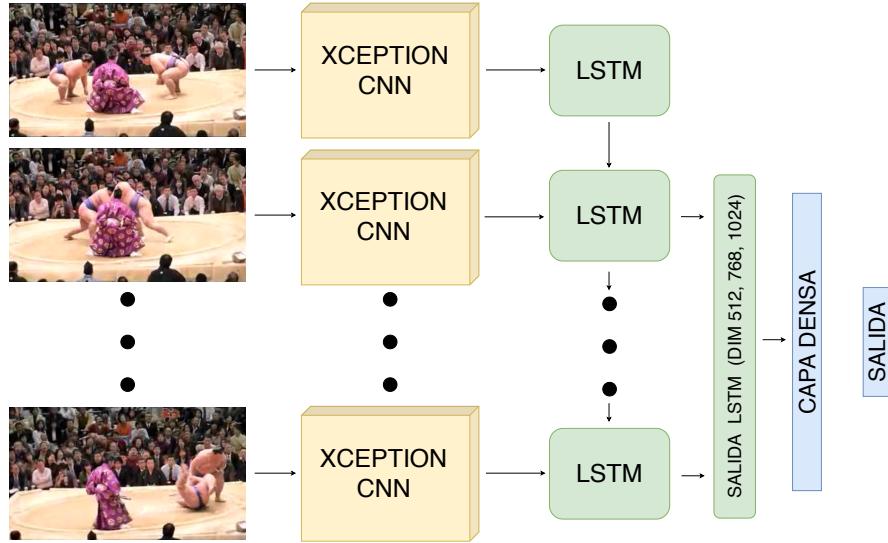


Figura 3.7: Extractor de características propuesto, junto con las capas completamente conectadas para el entrenamiento. Una vez el modelo ha sido preentrenado en el conjunto UCF-101, se eliminan las capas completamente conectadas. Por tanto, nuestro extractor de características final está compuesto por réplicas de la red Xception para cada uno de los fotogramas de entrada, y una red LSTM con tamaño de salida 512, 768 o 1024, que será lo que conoceremos como descriptor del fragmento de vídeo.

3.3.2. Arquitectura detectora de anomalías completa

Una vez ha sido preentrenado el extractor de características, procedemos al entrenamiento del modelo completo. Manteniendo estática la red extractora y eliminando toda la parte completamente conectada de dicho modelo, calculamos los descriptores para los fragmentos de vídeo. Con dichos fragmentos, haciendo la interpolación que proponen en el modelo original, obtenemos la representación de tamaño fijo para cada vídeo del conjunto de datos. Con dichas representaciones, entrenamos un clasificador completamente conectado utilizando la función de pérdida multi-instancia propuesta en el artículo.

Una vez tenemos el clasificador completamente conectado, construimos el modelo final. Dicho modelo está compuesto por el extractor de características hasta la salida de la red LSTM, seguido por la red completamente conectada.

Experimentación y resultados obtenidos

En este capítulo describiremos la experimentación realizada, así como los resultados obtenidos. Comenzaremos describiendo los aspectos de implementación y las bibliotecas utilizadas. Después, pasaremos a comentar cómo hemos repetido la experimentación original del artículo, y finalmente la experimentación de nuestra propuesta.

4.1. Aspectos de implementación

La implementación del trabajo ha sido realizada en Python 3.6. El código desarrollado para el trabajo, el cual permite replicar la experimentación llevada a cabo, se encuentra disponible en el repositorio <https://github.com/flluque1995/tfm-anomaly-detection>. Se proporcionan los modelos preentrenados para el experimento en el que se utiliza una representación de tamaño 1024. El resto de modelos no se han proporcionado porque la experimentación es similar en todos los casos, y hemos preferido incluir sólamente el mejor de los modelos.

Se utiliza como base para la implementación original el código localizado en https://github.com/ptirupat/AnomalyDetection_CVPR18. En dicho repositorio se ofrece una implementación del modelo, así como el archivo comprimido de la arquitectura original entrenada. No obstante, no está disponible el código necesario para entrenar de nuevo el sistema. Además, el autor del artículo ofrece el código en <https://github.com/WaqasSultani/AnomalyDetectionCVPR2018>.

Para que la comparación entre nuestro modelo y el original fuese más justa, se ha optado por repetir la experimentación llevada a cabo por los autores, para evitar así posibles diferencias en el entrenamiento de los modelos. Esto se debe a que no podemos garantizar que estamos ejecutando exactamente los mismos pasos que los autores originales a partir del código que proporcionan, ya que no está completo. Por ejemplo, en ninguno de los dos repositorios

se proporciona el código para la extracción de características, así que no podemos asegurar que los resultados que obtenga el modelo original a partir de las características extraídas por nosotros sean idénticos a los que se obtuvieron originalmente. A pesar de ello, incluimos también sus resultados, los cuales muestran que el modelo entrenado por nosotros no tiene exactamente el mismo comportamiento que el suyo.

De esta manera, tendremos dos versiones del modelo original, una basada en el modelo entrenado que ofrecen los autores, y otra entrenada completamente por nosotros. En cuanto a la implementación y entrenamiento completos del modelo original, se han utilizado las dos fuentes anteriores para replicar la experimentación de la forma más fiable posible. No obstante, debido a que había gran parte del código replicado entre ambos repositorios, y algunas partes estaban implementadas de forma poco eficiente, se han realizado algunos cambios en partes del código respetando su funcionamiento original. No se ha utilizado directamente el código original del artículo por la imposibilidad de replicar algunas de las etapas. Por ejemplo, hay partes del sistema escritas en MATLAB, o que necesitan del uso de herramientas externas. La implementación que hemos realizado permite repetir por completo la experimentación utilizando Python exclusivamente, y sin necesidad de recurrir a fuentes externas.

Para el tratamiento de los datos en formato de vídeo se ha utilizado la librería OpenCV [68]. OpenCV es una librería de código abierto pensada para el desarrollo de aplicaciones que se basan en la visión por computador. En particular, hemos utilizado los módulos que permiten la lectura de vídeos desde disco a memoria. Esta librería carga los vídeos como una lista de matrices numéricas, las cuales representan cada uno de los fotogramas que componen la secuencia de vídeo. Para el manejo de dichas matrices, hemos hecho uso de la librería de cálculo numérico NumPy [69, 70]. Este paquete ofrece una interfaz para operar de forma eficiente con colecciones estructuradas de números (*N-dimensional arrays*). Debido a que Python es un lenguaje orientado a la flexibilidad y la facilidad de uso, tiene como contrapartida una pérdida importante de rendimiento cuando se realizan operaciones aritméticas. Por este motivo, NumPy resulta ser una herramienta fundamental para el desarrollo de aplicaciones con alto coste computacional. Además, para la gestión efectiva de los archivos de anotaciones, los cuales se proporcionan en formato CSV, se ha hecho uso de Pandas [71]. Esta librería permite el manejo de datos estructurados en tablas, utilizando para ello estructuras conocidas como *DataFrames*.

Para la implementación de los modelos de redes neuronales se ha utilizado la librería Keras [66]. Esta biblioteca ofrece una interfaz de alto nivel para la implementación de modelos de aprendizaje profundo, utilizando por debajo la librería TensorFlow [72]. El uso de estos dos paquetes software facilita enormemente el desarrollo y validación de modelos, a la par que ofrece cierta flexibilidad. Esto permite la realización de pruebas sobre arquitecturas con cierta complejidad sin necesidad de escribir todo el sistema a bajo nivel, sino simplemente combinando neuronas de diversos tipos organizadas en capas. Además, el uso de TensorFlow permite la ejecución de los modelos en arquitecturas de GPU, en lugar de CPU. Debido a que las redes neuronales necesitan operar con matrices de grandes dimensiones, la ejecución de estos modelos exige realizar un número muy elevado de operaciones aritméticas. La ejecución de dichas operaciones es computacionalmente muy costosa, y la capacidad de utilizar el procesamiento en paralelo de las tarjetas gráficas hace que los tiempos de ejecución de los modelos se vean reducidos enormemente.

En cuanto a la arquitectura hardware en la que se han ejecutado los modelos, las ejecuciones se han llevado a cabo en un nodo de computación que dispone de tarjetas gráficas NVIDIA Tesla V100 de 32 GB de memoria RAM gráfica. Todos los modelos han sido ejecutados utilizando una sola unidad gráfica al mismo tiempo.

4.2. Evaluación de los modelos

Según el trabajo original, el modelo se debe evaluar utilizando el criterio a nivel de fotograma. Las anotaciones originales especifican cuatro valores, que corresponden al comienzo y final de la primera y la segunda anomalía presentes en el vídeo, respectivamente. Si hay una sola anomalía en la secuencia, los dos últimos valores son -1. Si no hay anomalía presente, es decir, el vídeo es normal, los cuatro valores son -1. Si hay más de dos fragmentos anómalos en el vídeo, sólo se indican los dos primeros. Para el cálculo de los resultados, se generan vectores de etiquetas con tantos elementos como fotogramas totales en el conjunto de datos.

En la experimentación original sólo se muestran dos métricas para la evaluación de los modelos; el área bajo la curva ROC, y la tasa de falsa alarma para vídeos normales. Aunque estas dos métricas son ciertamente relevantes para el problema que nos ocupa, consideramos que el uso de estas dos

medidas únicamente ofrece una comparación pobre entre la capacidad de predicción de los modelos. Por tanto, en este trabajo se calcularán algunas métricas más:

- ROC y AUC: Dan una medida bastante adecuada de la capacidad de clasificación general del modelo, al representar gráficamente la tasa de verdaderos y falsos positivos obtenidos bajo distintos umbrales de clasificación.
- Curva PR y precisión media: Esta curva, aunque es menos empleada que la anterior, ofrece también información relevante sobre los clasificadores. Muestra el compromiso entre la tasa de verdaderos positivos y la capacidad predictiva del modelo para la clase positiva. La precisión media se calcula como la media ponderada de los valores de precisión en distintos umbrales de la gráfica anterior.
- EER: Mide el punto en el cual la tasa de verdaderos y falsos positivos coincide. Se considera que un modelo es más potente cuanto menor es dicho umbral.
- Matriz de confusión para el umbral de clasificación 0.5: Dado que a priori no tenemos conocimiento del ámbito de aplicación del método, optamos por tomar como 0.5 la probabilidad que marca si un fotograma se considera, o no, anómalo. Utilizando dicho umbral, se calcula la matriz de confusión sobre el conjunto de test. Dicha matriz contiene el número de verdaderos negativos, falsos negativos, falsos positivos y verdaderos positivos.
- Exactitud: Aunque no es una métrica especialmente informativa cuando se presenta un problema no balanceado, se suele añadir en gran cantidad de trabajos, por lo que hemos decidido incluirla. Se define como la tasa de fotogramas correctamente clasificados respecto del total.
- Puntuación F_1 para la clase positiva: Se define como la media armónica entre la precisión y el ratio de verdaderos positivos. Da una intuición de cómo de bien se comporta el modelo para dicha clase, ya que esta métrica se ve fuertemente penalizada si el modelo tiene un mal comportamiento en términos de falsos positivos o negativos.

Además de las métricas anteriores, las cuales se calculan todas a nivel de fotograma, calcularemos dos medidas más a nivel de vídeo. Estas dos medidas son el porcentaje de vídeos anómalos y normales para los cuales se

genera al menos una predicción de anomalía. La primera de estas medidas nos dará una idea el número de vídeos que hacen saltar la alarma, de forma que un valor alto para esta medida nos indicará que el modelo reconoce bien los vídeos anómalos, aunque posiblemente la localización temporal de dicha anomalía no sea del todo precisa. La segunda de las medidas nos dará una idea de la cantidad de vídeos normales en los que ha saltado la alarma, los cuales son claros falsos positivos. Los fotogramas considerados falsos positivos pueden ser provocados por una mala localización en los extremos de una anomalía real. Estos, en cambio, marcan falsos positivos seguros, ya que ningún fotograma de estos vídeos debería ser marcado como anómalo.

4.3. Reproducción de la experimentación original

En esta sección estudiaremos cómo hemos reproducido la experimentación original. Comenzamos describiendo la extracción de los descriptores de vídeo.

4.3.1. Extracción de características

El primer paso consiste en la extracción de características de los vídeos. Debido a que el extractor de características está preentrenado y no se reentrena durante el entrenamiento del clasificador, se pueden extraer los descriptores de vídeo a priori para no tener que recalcularlos en cada etapa de entrenamiento de la red final.

Para la extracción se ha utilizado la implementación del modelo C3D disponible en <https://github.com/adamcasson/c3d>. Dicha implementación es una adaptación del modelo original de C3D [61] para Keras, ya que originalmente el modelo estaba escrito en Caffe. Los pesos del modelo preentrenado en el conjunto de datos Sports-1M están disponibles para la descarga en el repositorio anterior.

Utilizando dicho modelo, para cada vídeo del conjunto de datos de entrenamiento, se divide la secuencia completa en intervalos de 16 fotogramas sin solapamiento, descartando los últimos si es necesario. Para cada fotograma, se redimensiona al tamaño 128×171 , y se recorta un cuadrado centrado de tamaño 112×112 . Dicha transformación se realiza para adecuar los fotogramas a la entrada de la red sin distorsionar mucho la imagen (cuando se hace una redimensión a tamaño cuadrado directamente, los objetos aparecen estirados en la imagen). Además, se resta la media de entrenamiento que viene

especificada en el modelo C3D (uno de los pesos que se descargan es el vector de medias para normalizar). Una vez se tienen los fragmentos de tamaño $16 \times 112 \times 112 \times 3$, se hacen pasar a través de la red, y se extrae la salida de la primera capa completamente conectada que presenta el modelo, la cual tiene un tamaño de 4096 elementos. Dicha salida, que será la representación del segmento, se normaliza utilizando la norma L_2 .

Una vez completado este proceso, se tiene una matriz de tamaño $k \times 4096$, donde k es el número de fragmentos de 16 fotogramas que se han podido extraer del vídeo. Para obtener la representación final del vídeo, la cual tiene que estar compuesta por 32 segmentos, se agrupan consecutivamente y de forma lo más equitativa posible los fragmentos de 16 fotogramas y se calcula la media de éstos. De nuevo, se normaliza el resultado, ya que en general la media de un conjunto de vectores de norma 1 no tiene por qué conservar dicha norma. La primera normalización se realiza para garantizar que unos vectores no resultan más influyentes que otros en la media, y la segunda para trabajar con entradas normalizadas en la red neuronal. Tras este proceso, se obtiene la representación final de cada vídeo, que tiene un tamaño de 32×4096 .

4.3.2. Entrenamiento del clasificador

Una vez que se han extraído las características de los vídeos de entrenamiento, se procede al aprendizaje del clasificador. El clasificador es una red completamente conectada compuesta por tres capas, dos de ellas ocultas, de 512 y 32 neuronas respectivamente, y la capa de salida, con una única neurona. La función de activación de las dos capas ocultas es la función lineal rectificada (ReLU), y la de la capa de salida es la función sigmoide, para conseguir una salida entre 0 y 1, que sirva como “probabilidad” de anomalía.

Entre cada pareja de capas se añade una capa de *Dropout* con un ratio de 0.6, lo que hace que durante el entrenamiento, en cada época, se desactiven aleatoriamente el 60% de las neuronas de la capa. Esta estrategia se utiliza para evitar que el conocimiento de la red recaiga sobre unas pocas neuronas, ya que al desactivar partes de la red aleatoriamente durante el entrenamiento se obliga a repartir el conocimiento entre distintas partes de la red.

Además, se añade a la función de pérdida explicada en el capítulo anterior un término de regularización L_2 para todas las capas. Esto se hace para evitar que los pesos de la red crezcan descontroladamente, lo cual suele tra-

ducirse en una pérdida de rendimiento y aparición de sobreajuste. Los pesos que se utilizan para los términos de regularización son 0.00008 para las restricciones dispersas y de suavidad temporal, y 0.001 para los pesos de las capas.

El entrenamiento de la red se realiza durante 20000 épocas. En cada época se seleccionan aleatoriamente 30 vídeos normales y 30 vídeos anómalos para formar el lote de entrenamiento. El optimizador utilizado es Adagrad, con una tasa de aprendizaje inicial de 0.001.

4.3.3. Inferencia sobre el conjunto de test

Una vez se tiene el modelo entrenado, la forma de generar predicciones sobre el conjunto de test consiste en realizar el mismo preprocesado a cada vídeo, clasificarlo con el modelo completo (extracción de características en fragmentos de 16 fotogramas, transformación en 32 segmentos y clasificación de dichos segmentos), y extrapolar los 32 resultados obtenidos linealmente, para conseguir una predicción para cada fotograma. Dado que se obtiene un valor entre 0 y 1 como salida del modelo, se considera que un fotograma es anómalo si el valor asignado a dicho fotograma es superior a 0.5.

4.4. Experimentación propia

En esta sección describiremos la experimentación propia realizada. En primer lugar, comentamos cómo se ha llevado a cabo el entrenamiento del extractor de características.

4.4.1. Entrenamiento del extractor Xception-LSTM

Como comentamos anteriormente, el entrenamiento del extractor de características se ha realizado sobre un conjunto de datos de mayor tamaño, y el modelo obtenido se ha congelado y utilizado para extraer las representaciones de los vídeos de nuestro conjunto objetivo.

El entrenamiento del extractor se ha realizado sobre el conjunto de datos UCF-101, como describimos anteriormente. Para controlar la evolución del modelo y comprobar si sobreajusta, se utilizan como conjuntos de entrenamiento y validación los propuestos por los autores del conjunto de datos como primeras particiones. Para que la comparación entre modelos sea justa,

el conjunto de datos viene dividido en entrenamiento y test tres veces. De esta forma, se exige que los modelos se entrenen y evalúen en tres ocasiones, reduciendo así la posible influencia del azar. En nuestro caso, utilizaremos la primera de las tres divisiones para entrenar nuestro extractor. Usando la partición de test como conjunto de validación, podemos observar cómo se comporta el extracto sobre datos que no ha observado mientras se está entrenando, dando así una idea de su capacidad de generalización. Utilizando esta división, se entrena el clasificador sobre un conjunto de 9537 vídeos, y se evalúa sobre un conjunto de 3783. Es posible que la división que hemos utilizado no sea la más adecuada, ya que el conjunto de validación es relativamente grande. En nuestro caso, no necesitamos un conjunto de validación que garantice una comparación justa entre nuestro modelo y el resto de modelos que resuelven este problema, si no que queremos hacernos una idea del funcionamiento correcto de nuestro extracto. Es posible que con un conjunto de entrenamiento de mayor tamaño, a cambio de uno menor de validación, consiguiésemos características de mayor calidad, al tener más diversidad de entrenamiento. No obstante, no hemos entrado en estudiar este detalle en más profundidad porque la división por defecto ya deja un conjunto de datos de entrenamiento de tamaño aceptable, suficiente para la experimentación que se afronta en este trabajo.

Como ya comentamos anteriormente, para entrenar el extracto de características, modificamos la última parte de la red para resolver el problema de clasificación en este conjunto de datos. Concretamente, a la arquitectura extractora (compuesta por los módulos Xception para el tratamiento de imagen más la capa LSTM para el aprendizaje temporal), se añaden dos capas ocultas, y la capa de salida, que tiene 101 neuronas (tantas como clases en el conjunto de datos). Por tanto, tenemos la siguiente estructura final:

- Módulo Xception, que acepta imágenes de tamaño $299 \times 299 \times 3$, y devuelve un descriptor de las mismas de tamaño 2048. Dado que trabajamos con 16 fotogramas, utilizando el módulo TimeDistributed de Keras podemos tener 16 copias de la red en paralelo, para que la salida sea una serie temporal de 16 instantes de tiempo y 2048 características por instante. Esta parte de la red está entrenada sobre ImageNet y no se reentrena.
- Capa recurrente LSTM. Acepta como entrada una serie temporal de 16 instantes de tiempo y 2048 características, y devuelve un descriptor de la serie de tamaño 512, 768 o 1024, dependiendo del tamaño

del descriptor que queramos extraer. Esta parte de la red se inicializa aleatoriamente y se entrena en esta etapa.

- Clasificador de capas densas. Esta parte de la red, formada por tres capas, procesa el descriptor temporal y realiza la clasificación final. Al igual que la parte recurrente, se inicializa aleatoriamente y se entrena en esta etapa. Todo este módulo se desechará tras el entrenamiento, conservándose de esta manera la parte extractora de características, exclusivamente. El tamaño de las dos capas ocultas dependerá del tamaño del descriptor (por ejemplo, en el caso del descriptor de tamaño 1024, las capas densas tienen tamaño 512 y 128). La última capa tiene siempre 101 neuronas, tantas como clases tenemos en el conjunto de datos.

Para construir los ejemplos del conjunto de datos, debido a que los vídeos originales tienen tamaños y duraciones distintas, se han escalado los fotogramas para que tengan tamaño 299×299 (en nuestra experimentación no se ha tenido en cuenta la distorsión de la misma forma que se tiene en C3D), y se han muestreado los vídeos para obtener fragmentos de 16 fotogramas equiespaciados entre el principio y el final de cada vídeo. Obtenemos por tanto una representación para cada secuencia de $16 \times 299 \times 299 \times 3$. Para el preprocesado de cada fotograma, Keras aporta una función que normaliza la entrada para adecuarla a cómo se entrenó el modelo Xception. Dicha función, a grandes rasgos, realiza una normalización del valor de cada píxel dividiendo por la desviación típica y restando la media de los valores de los píxeles del conjunto ImageNet.

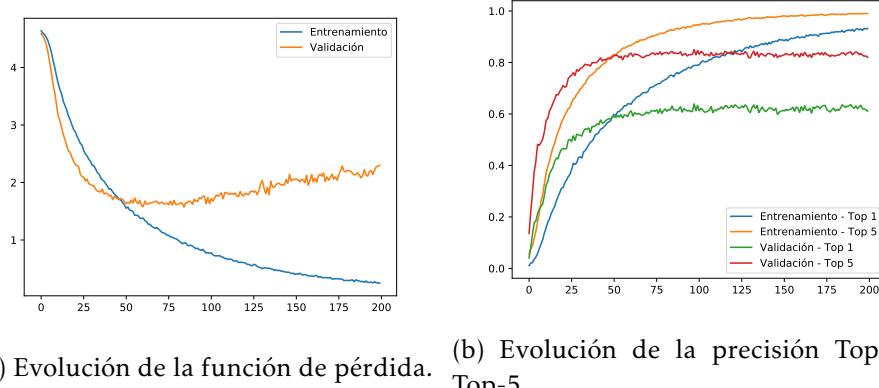
Para el entrenamiento del modelo se utiliza el optimizador Adam con una tasa de aprendizaje de 10^{-5} , y un decaimiento de 10^{-6} (este optimizador reduce la tasa de aprendizaje tras cierto número de épocas, para afinar el resultado en las últimas etapas del entrenamiento). La función de coste a optimizar es la entropía categórica, y se entrena el modelo durante 200 épocas. Para conservar el mejor modelo encontrado hasta el momento, se utiliza la funcionalidad ModelCheckpoint de Keras, que te permite especificar una métrica y el período de comprobación, y guarda una copia del modelo en el momento de entrenamiento en el que se comprueba si la métrica es la mejor obtenida hasta el momento. Utilizamos como métrica a observar la precisión obtenida sobre el conjunto de validación. Observamos una métrica en el conjunto de validación para que el sobreajuste no nos induzca a pensar que el modelo tiene muy buen comportamiento, a pesar de una capacidad de generalización pobre.

Durante el entrenamiento, se han recogido tres métricas distintas, que suelen utilizarse en la evaluación de problemas de clasificación con una cantidad de clases tan amplia; el valor de la función de coste, que indica si el entrenamiento se está llevando a cabo correctamente, la precisión “Top-1”, que indica el porcentaje de ejemplos correctamente clasificados, y la precisión “Top-5”, que indica el porcentaje de ejemplos cuya clase real está entre las cinco clases más probables predichas por el modelo. Estas dos últimas métricas son casos particulares de la precisión “Top-k”, que es muy útil para evaluar problemas con gran cantidad de clases. En muchos casos, las clases representadas en los conjuntos de datos de este tipo tienen un gran solapamiento porque representan conceptos similares. En las predicciones de los modelos aparecen varias clases con una probabilidad alta, muy parecida para todas ellas, entre las que se suele encontrar la clase real. La métrica “Top-k” soluciona en parte el hecho de que el modelo haya seleccionado la clase correcta como una de las más probables, pero no como la más probable de todas. Claramente, es un error mucho menos grave predecir de forma casi equiprobable las clases “Montar a caballo” y “Carrera de caballos” (ambas presentes en el conjunto UCF-101), pero poner ligeramente por encima la incorrecta, que asignar con alta probabilidad la clase “Tocar la guitarra” alta a un ejemplo de “Montar a caballo”. Es por esto por lo que se considera que la métrica “Top-k” para $k \geq 3$ es usualmente más justa que la métrica “Top-1” para problemas con tantas clases.

En las gráficas de la figura 4.1 se puede observar la evolución de estas tres métricas durante el entrenamiento del modelo de tamaño 1024 de representación. No mostramos la evolución de los otros dos modelos debido a que el razonamiento sobre los mismos es similar y no aporta nueva información.

A la vista de los resultados que pueden observarse en las mismas, hemos obtenido un extracto de características de bastante calidad. Observamos cómo a partir de las 100 épocas el modelo comienza a sobreajustar, ya que en la gráfica de la función de coste el valor en el conjunto de entrenamiento sigue descendiendo, pero en el de validación se estanca y comienza a subir. Además, en la gráfica de la precisión se produce un estancamiento de los valores obtenidos.

Gracias al `ModelCheckpoint` que comentamos previamente, hemos conservado el estado del modelo en la época 100 para nuestro extracto de características. Hemos considerado que en dicho punto se obtiene un modelo de calidad a partir de las métricas que hemos calculado. En el punto de guarda-



(a) Evolución de la función de pérdida. (b) Evolución de la precisión Top-1 y Top-5.

Figura 4.1: Evolución de la función de pérdida y precisiones para los conjuntos de entrenamiento y validación utilizando el extractor de características de tamaño 1024.

do anterior, a las 80 épocas, el modelo tiene un comportamiento ligeramente peor en términos de precisión tanto Top-1 como Top-5, y unos valores de la función de coste muy similares. El siguiente punto de guardado, a las 120 épocas, ya se encuentra en la fase en la que el modelo sobreajusta, por lo que nos hemos quedado con el modelo intermedio.

Tras el entrenamiento de los modelos extractores, obtenemos la siguiente tabla de precisiones:

Dimensión del descriptor	Precisión Top-1	Precisión Top-5
512 elementos	57.63 %	82.42 %
768 elementos	62.98 %	84.67 %
1024 elementos	63.27 %	84.66 %

Tabla 4.1: Resultados Top-1 y Top-5 de los tres extractores sobre UCF-101

Como podemos comprobar, para las tres dimensiones obtenemos un modelo de bastante calidad. El modelo C3D original muestra en su experimentación [61, Figura 2] que su modelo llega a una precisión Top-1 de 45 % cuando se entrena como una red neuronal completa (al igual que hemos hecho nosotros). Después, muestran los resultados de su modelo final, que emplea las características de C3D pero utiliza una SVM para la clasificación final, y cuyos resultados son significativamente mejores, con una precisión Top-1 de

más del 80%. No obstante, como estamos interesados en comparar los extractores de características, creemos que la comparación justa debe ser con el modelo neuronal completo. Por los resultados obtenidos, podemos concluir que nuestro extractor es más potente que el que ellos proponen, al menos en cuanto a clasificación en UCF-101 se refiere.

Comparando los resultados de nuestros tres extractores, parece claro que existe una mejoría cuanto mayor es el tamaño del vector de características. Dicha mejora no es tan notable entre el extractor de dimensión 768 y el de tamaño 1024, cuyos resultados son muy similares (especialmente en términos de precisión Top-5). No obstante, sí que se observa una mejora sustancial entre estos dos modelos y el más pequeño.

4.4.2. Extracción de características

Para la extracción de características de los vídeos del conjunto UCF-Crime, seguimos la misma política que en el trabajo original. Dividimos el vídeo en fragmentos de 16 fotogramas sin solapamiento, preprocesamos los fotogramas para llevarlos a tamaño 299×299 , y utilizamos la función proporcionada por Keras de preprocesado para Xception.

Utilizando la red entrenada según el apartado anterior sin las capas densas, conseguimos para cada vídeo una representación de tamaño $k \times 1024$, con k igual al número de fragmentos no solapados de 16 fotogramas. Agrupando dichos fragmentos en 32 grupos y tomando la media de los fragmentos de cada grupo, obtenemos la representación final del vídeo formada por 32 fragmentos, cada uno de ellos con un vector de características de tamaño 1024.

4.4.3. Entrenamiento del clasificador

Una vez tenemos las características extraídas, aplicamos una estrategia análoga a la del modelo original para el entrenamiento del clasificador. El clasificador es de nuevo una red neuronal completamente conectada, pero de tamaño menor a la empleada por el modelo original. Tenemos dos capas ocultas, de tamaño 512 y 64, respectivamente, y la neurona de salida. La disminución del tamaño de las capas ocultas viene justificada por la reducción del tamaño del descriptor, que ha pasado a tener la mitad del tamaño original.

Además, se ha reducido la tasa de desactivación de las capas de *Dropout* de 0.6 a 0.4, ya que hemos considerado que en el modelo original se tomaba un valor excesivo, el cual puede hacer también que el rendimiento del clasificador se degrade y el entrenamiento sea mucho más lento.

Se han ajustado también los valores de ponderación de los términos de la función de pérdida. Se ha aumentado el valor el regularizador del núcleo a 0.01, y se han reducido los multiplicadores de las restricciones temporales y dispersas a la mitad (de 0.00008 a 0.00004). Estos ajustes se han hecho tras estudiar los primeros entrenamientos del modelo y observar que cometía demasiados falsos negativos. Esto podía ocurrir porque las restricciones de dispersión fuesen demasiado fuertes, así que hemos mitigado ligeramente ese efecto reduciendo la importancia del término.

El optimizador utilizado en nuestro caso es de nuevo Adagrad, aunque hemos aumentado la tasa de aprendizaje a 0.2 al principio. Se ha observado que el entrenamiento es bastante lento al principio, y esta modificación mejora este mal comportamiento inicial.

4.5. Resultados obtenidos

En esta sección, mostramos los resultados obtenidos en el experimento final. En primer lugar, mostramos las matrices de confusión para el umbral de decisión 0.5 de ambos modelos:

Modelo	TN	FP	FN	TP
Original - preentrenado	902433	125044	49699	34632
Original - replicado	783342	244135	43699	40632
Xception-LSTM - 1024	875515	151962	44145	40186
Xception-LSTM - 768	908518	118959	52874	31457
Xception-LSTM - 512	914259	113218	60661	23670

Tabla 4.2: Matrices de confusión para los modelos entrenados

En la tabla podemos observar varios detalles que merece la pena comentar. En primer lugar, podemos observar cómo el modelo original entrenado por los autores (marcado aquí como preentrenado) y el modelo original replicado por nosotros no obtienen resultados equivalentes. Nuestra réplica consigue un mayor número de verdaderos positivos, con un aumento de 6000

fotogramas anómalos detectados, pero a cambio de aumentar significativamente el número de falsos positivos. Este es el motivo por el que hemos decidido replicar la experimentación, ya que normalmente es muy difícil replicar las condiciones de entrenamiento de un modelo, y por tanto la comparativa con los resultados originales puede no ser del todo justa. El modelo replicado por nosotros ha sido entrenado en unas condiciones similares a las que hemos impuesto a nuestra propuesta, por lo que la comparativa es, desde nuestro punto de vista, más justa con dicho modelo.

Comparando nuestras propuestas entre sí, podemos observar cómo el tamaño de la representación afecta significativamente a los resultados que obtiene el modelo. Cuanto mayor es el vector de características, más fotogramas anómalos se detectan, a costa de cometer también más falsos positivos. No obstante, el aumento de falsos positivos no es tan notable como el aumento de fotogramas bien clasificados, por lo que consideramos que nuestro modelo mejora con el aumento del tamaño de la representación.

Resulta especialmente notable comparar el aumento de falsos positivos entre los modelos. Mientras que para los modelos de 512 y 768 características, el aumento de falsos positivos es pequeño (de hecho, en términos absolutos, el aumento de falsos positivos es menor que el aumento de verdaderos positivos, a pesar del desbalanceo de clases), cuando observamos el modelo de tamaño 1024 este problema se dispara. Convendría estudiar por qué se ha producido dicho aumento.

En cuanto a la comparativa entre nuestra propuesta y el modelo original, si nos comparamos con nuestra experimentación, el modelo propuesto de mayor tamaño de representación es significativamente mejor que la propuesta original. El número de verdaderos positivos es prácticamente el mismo, pero el número de falsos positivos se ha reducido drásticamente. Esto indica que las características que hemos extraído permiten diferenciar mejor los fotogramas normales de los anómalos que las que extrae el modelo C3D. Si nos comparamos con la propuesta original entrenada por los autores originales, obtenemos un aumento considerable en el número de verdaderos positivos (detectamos unos 6000 fotogramas positivos más), pero a costa de un aumento también significativo de falsos positivos. En este caso, probablemente, la decisión entre escoger un modelo u otro si nos fijamos exclusivamente en las matrices de confusión vendría justificada por los requerimientos del problema real a resolver. En este caso, el detectar un mayor número de positivos es recomendable, ya que estamos hablando de un sistema de alarma. Cometer

un falso positivo será menos grave que cometer un falso negativo, ya que en el caso de falso positivo simplemente se avisa de algo que en realidad no está ocurriendo, pero en el caso del falso positivo podemos ignorar una situación de peligro que sí interesaría detectar.

Además de las matrices de confusión, mostramos a continuación las curvas ROC y las curvas PR:

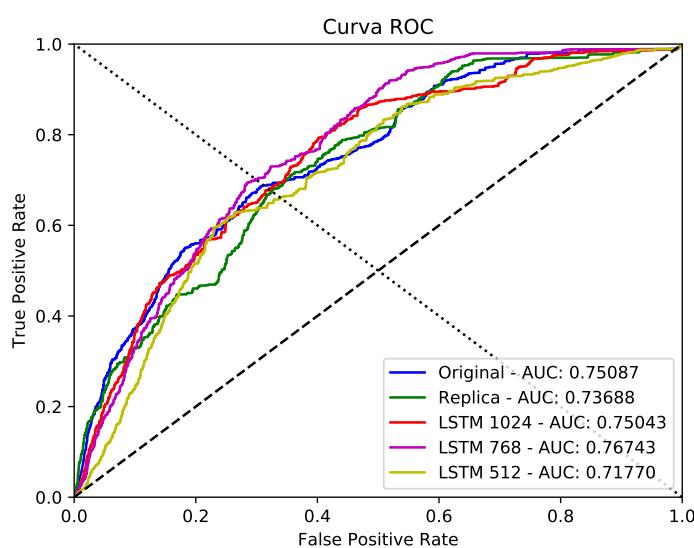


Figura 4.2: Curvas ROC de los modelos

A la vista de las curvas ROC, podemos concluir que todos los modelos tienen un comportamiento similar. En principio, los dos modelos de peor calidad en cuanto a la curva ROC se refiere son la réplica del modelo original y el modelo Xception-LSTM con tamaño de 512 características. Estas dos curvas son dominadas por las otras 3 prácticamente en todo el gráfico. Por otro lado, la curva obtenida por el modelo Xception-LSTM de tamaño 768 es claramente la dominante en gran parte del gráfico. Para tasas bajas de falsos positivos dicha ventaja no es tan notable, pero rápidamente pasa a ser el mejor modelo y domina al resto de curvas en el resto del gráfico, alcanzada sólamente por la representación de 1024 elementos para una tasa de falsos positivos en torno al 0.4.

Otro detalle que podemos observar es que los modelos basados en convolución exclusivamente tienen mejor comportamiento que nuestros modelos cuando hablamos de tasas bajas de falsos positivos. Tanto el modelo origi-

nal como la réplica se mantienen como mejores modelos cuando hablamos de una tasa de falsos positivos por debajo de 0.1. No obstante, a partir de ese punto, nuestras propuestas alcanzan un rendimiento similar a las propuestas originales, y las superan rápidamente para el resto del gráfico.

Si nos fijamos en términos de AUC, sorprendentemente el mejor modelo encontrado es el que utiliza la representación de tamaño 768. Aunque en la matriz de confusión nos pareció mejor el modelo de tamaño 1024, en términos de AUC tenemos un mejor comportamiento por parte del modelo de tamaño medio. En particular, supera por más de 1.5 puntos porcentuales tanto al modelo original como a nuestra propuesta de mayor tamaño, los cuales consiguen unos resultados muy similares. Estamos hablando de una mejora relativamente amplia, por lo que este modelo podría ser el más adecuado en algunos contextos. Además, el EER, que es el punto en el que la curva ROC se corta con la línea de puntos diagonales, muestra también que el mejor modelo es la LSTM de tamaño 768. A continuación, con un resultado muy similar entre ellas, están las dos convolucionales completas y el modelo LSTM de tamaño 1024. Finalmente, el modelo que peor comportamiento tiene en este caso es de nuevo la LSTM de tamaño 512.

Pasamos a mostrar las curvas PR:

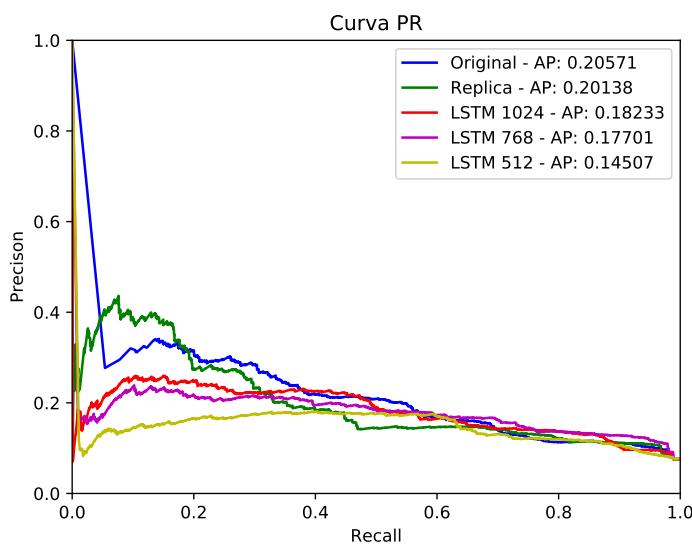


Figura 4.3: Curvas PR de los modelos

En esta comparativa encontramos resultados muy dispares. Por un lado,

si nos fijamos en la parte izquierda de la gráfica, tenemos que la réplica del modelo original es la que mayor precisión tiene, pero ahí estamos hablando de tasas de verdaderos positivos bajas (esta métrica recibe en inglés el nombre de *Recall*, y es el nombre que se muestra aquí). También tenemos por encima en este caso al modelo original. Conforme vamos avanzando por el gráfico hacia niveles mayores de *recall*, la réplica pasa a perder mucha precisión, y rápidamente se convierte en el peor de los modelos. Algo similar le ocurre al modelo original, el cual acaba siendo superado por el modelo de tamaño 1024 y el de tamaño 768. No obstante, la pérdida de rendimiento en este caso es mucho más gradual que en la réplica.

Nuestros tres modelos, por el contrario, tienen un comportamiento similar durante todo el gráfico. Aunque no llegan a tener niveles de precisión muy altos en ningún punto, sí que se mantiene la misma de forma más o menos constante. Llegan a superar a los modelos originales en una gran parte del gráfico. No obstante, debido al buen rendimiento del modelo convolucional en tasas bajas de verdaderos positivos, cuando miramos la precisión media tanto el original como la réplica quedan por encima de todas nuestras propuestas.

Este buen comportamiento al principio significa que los modelos convolucionales consiguen diferenciar más claramente los comportamientos claramente anómalos que los modelos Xception-LSTM. Es decir, para aquellos ejemplos en los que es clara la anomalía, el modelo convolucional suele responder de forma más segura, por lo que clasifica correctamente un porcentaje importante de fotogramas de este tipo. Por otro lado, cuando empiezan a aparecer fotogramas de clasificación más difícil, los modelos Xception-LSTM pasan a comportarse mejor que los modelos convolucionales puros.

Pasamos a mostrar y analizar la tabla de métricas comparativas:

Modelo	Exactitud	AUC	F_1	EER	AP
Original - preentrenado	0.8428	0.7508	0.2838	0.3119	0.2057
Original - replicado	0.7411	0.7369	0.2201	0.3253	0.2014
Xception-LSTM - 1024	0.8236	0.7504	0.2907	0.3221	0.1823
Xception-LSTM - 768	0.8455	0.7674	0.2681	0.2980	0.1770
Xception-LSTM - 512	0.8436	0.7177	0.2140	0.3388	0.1451

Tabla 4.3: Tabla de métricas comparativas de los modelos a nivel de fotograma

La primera métrica con la que nos encontramos es la exactitud. Aunque sabemos que esta métrica es poco relevante cuando nos encontramos ante un problema de clasificación muy desbalanceado, como el nuestro, es una métrica que se calcula normalmente, por lo que hemos decidido incluirla. Para esta métrica, tenemos el mejor comportamiento para el modelo LSTM de tamaño 768, aunque casi todos los modelos obtienen unos resultados similares. El modelo original replicado es claramente el peor de los cinco en esta métrica, el cual obtiene unos 10 puntos porcentuales menos que el resto.

En términos de AUC, como ya habíamos visto en la gráfica anterior, el mejor modelo es la propuesta de tamaño 768, que supera ampliamente al resto de modelos. En particular, obtiene mejores resultados que el propio modelo original en el artículo, lo cual abre la puerta a una posible publicación tras un estudio más exhaustivo. Es importante remarcar que nuestro extractor fue preentrenado en un conjunto que contiene menos información que el utilizado en el extractor original, por lo que este margen de mejora que hemos obtenido podría agrandarse más con los nuevos datos. En cuanto al resto de modelos, podemos observar cómo el modelo original consigue un resultado similar al obtenido por la propuesta de tamaño 1024. Ligeramente por debajo queda el modelo réplica, y por último el modelo de Xception-LSTM de tamaño 512.

La mejora que consideramos más importante derivada de nuestro estudio se produce sobre la métrica F_1 para el umbral de decisión 0.5. Esta mejora era previsible teniendo en cuenta los resultados que observamos sobre la matriz de confusión, pero aquí confirmamos que efectivamente se produce. En particular, hemos mejorado esta métrica con el modelo propuesto de tamaño 1024, que consigue la puntuación más alta de todos los modelos. Consideramos esta métrica especialmente relevante porque representa el comporta-

miento que tiene nuestro modelo respecto a la clase positiva, que es en la que más interesados estamos. Esto implica que, para el umbral de decisión fijado, el modelo Xception-LSTM de mayor tamaño se comporta mejor que el modelo original, lo cual apoya la hipótesis que teníamos de partida. Resulta curioso observar que, a pesar de ser el mejor modelo en términos de AUC por un margen importante, el modelo de dimensión 768 obtiene peores resultados en esta métrica. Esto pone de manifiesto la importancia de utilizar diversas métricas para comparar modelos, ya que el uso de una única medida, a no ser que estemos muy interesados en un comportamiento concreto, suele dar ideas erróneas.

Las dos últimas métricas que mostramos hemos podido observarlas previamente en las gráficas mostradas. La tasa de error igual (EER), la cual indica un mejor modelo cuanto menor sea el valor, pudimos observarla en la gráfica de las curvas ROC, ya que es el punto en el que se cruzaban las curvas con la diagonal de puntos. Como ya dijimos, el modelo de dimensión 768 es el mejor modelo en términos de esta métrica. El AP, que se mostraba en la leyenda de la gráfica PR, nos dice que los mejores modelos en términos de precisión media son los modelos convolucionales puros, debido a la precisión tan alta que tenían para las tasas de acierto bajas.

Por último, mostramos los resultados en términos de capacidad de clasificación de los modelos a nivel de vídeo. La capacidad de los modelos de detectar la presencia de una anomalía puede ser incluso más importante que la localización exacta de dicha anomalía dentro del vídeo. Aunque la anomalía no esté perfectamente delimitada, es decir, que los primeros o últimos segundos de la anomalía no queden precisamente delimitados, el hecho de generar la alarma cuando efectivamente se produce una situación anómala, y no hacerlo cuando nos encontramos ante un vídeo normal, es de gran importancia si se pretende implementar un sistema de estas características en un entorno real.

En la siguiente tabla se pueden observar los resultados de los modelos en términos de evaluación a nivel de vídeo:

Modelo	% Vídeos normales	% Vídeos anómalos
Original	13.33	64.89
Réplica	11.11	74.05
Xception-LSTM - 1024	15.55	77.86
Xception-LSTM - 768	12.59	72.52
Xception-LSTM - 512	8.15	71.76

Tabla 4.4: Porcentaje de vídeos normales y anómalos en los que se ha generado una alarma. Para los vídeos normales, un porcentaje mayor implica que se han generado más falsas alarmas en vídeos normales, y por tanto un peor rendimiento del modelo. Para los vídeos anómalos, por el contrario, un porcentaje más alto indica que se han detectado anomalías en un número mayor de vídeos, y por tanto mejores resultados.

Para estas métricas, podemos observar que nuestra propuesta es también de bastante calidad. Por un lado, se observa que el modelo original tiene un comportamiento bastante mejorable, especialmente en términos de vídeos anómalos sin etiqueta. En particular, no se generan alarmas en uno de cada tres vídeos, lo que significa que es un modelo bastante poco fiable para su uso en entornos reales. El modelo original replicado por nosotros mejora notablemente estos resultados, generando un menor número de alarmas en vídeos normales, y un aumento significativo de alarmas para vídeos anómalos. Concretamente, sólo se pierden alarmas en uno de cada cuatro vídeos. Sigue siendo un valor alto, pero significativamente mejor que el original.

Comparando entre si los resultados obtenidos por nuestros modelos, tenemos que cuanto mayor es el tamaño de representación, más información de anomalía se recoge. El aumento del tamaño del descriptor lleva acompañado un aumento importante del número de vídeos positivos en los que se genera una alarma. Como contrapartida, también se cometan un número mucho mayor de falsos positivos. Es especialmente notable el mal comportamiento del modelo de 768 características en este caso. Apenas mejora el porcentaje de vídeos detectados respecto al modelo de tamaño 512, y por otro lado comete casi tantas falsas alarmas como el modelo de tamaño 1024.

Entre el modelo de tamaño 1024 y el modelo original entrenado por nosotros, que son los dos que mejores resultados obtienen, la decisión entre cuál tiene mejor comportamiento es subjetiva. Nuestro modelo es capaz de detectar un mayor número de vídeos anómalos, pero a cambio de un aumento en

el número de falsas alarmas importante. Probablemente, si el modelo va a utilizarse en una aplicación real, estaremos más interesados en un modelo similar al nuestro, ya que interesaría detectar el mayor número de situaciones anómalas posible. El hecho de que ocurran falsos negativos es menos preocupante, ya que una alarma de este tipo servirá, probablemente, como método de aviso y no como toma de decisiones. Nos interesa, por tanto, que sea muy sensible a la clase positiva, y no tanto a la clase negativa, ya que las falsas alarmas van a ser procesadas a posteriori por un humano.

Conclusiones y trabajo futuro

5.1. Conclusiones

En este trabajo se ha realizado un estudio exhaustivo del uso del aprendizaje profundo para el análisis de multitudes en videovigilancia. En primer lugar, se ha llevado a cabo un análisis detallado del estado del arte, que ha resultado en la publicación de un artículo científico en la revista *Information Fusion*, y que lleva por título “Revisiting crowd behaviour analysis through deep learning: Taxonomy, anomaly detection, crowd emotions, datasets, opportunities and prospects” [73]. En dicho estudio, se ha propuesto una taxonomía que permite organizar los nuevos trabajos en una secuencia de pasos, de forma que los resultados de cada una de las etapas tienen una fuerte influencia en las etapas posteriores. Para la tercera de las etapas de la taxonomía propuesta, que corresponde a la fase de extracción de características, se han establecido las principales propiedades que se extraen de las secuencias de vídeo para el análisis de comportamientos en multitudes.

Además, se ha realizado una revisión bibliográfica exhaustiva de los modelos basados en aprendizaje profundo para la detección de anomalías en multitudes. En primer lugar, se han identificado las distintas subtareas que componen esta área, las cuales vienen determinadas por las diferentes fuentes que producen la anomalía. Para los tipos de anomalía identificados, se han recopilado los principales conjuntos de datos públicos y las principales métricas que se utilizan para evaluar la calidad de los modelos. Finalmente, se han resumido los diferentes trabajos que resuelven cada una de las subtareas identificadas utilizando aprendizaje profundo.

Para el apartado práctico del trabajo, se ha estudiado la eficacia del uso de características espacio-temporales extraídas con modelos de aprendizaje profundo para la detección de anomalías en vídeo. Concretamente, se ha experimentado sobre un modelo de detección de anomalías en multitudes que empleaba un extracto de características basado exclusivamente en re-

des neuronales convolucionales en tres dimensiones. Para dicho modelo, se ha sustituido el extractor de características por un compuesto de capas convolucionales y recurrentes. Nuestra hipótesis de partida defendía que las redes neuronales recurrentes iban a ser mejores extractores de características temporales que las redes convolucionales 3D.

A raíz de los resultados extraídos de la experimentación, hemos podido comprobar que en efecto el modelo combinado convolucional-recurrente tiene un mejor comportamiento que el modelo puramente convolucional para el análisis de secuencias de vídeo.

Por un lado, trabajando con el conjunto de datos UCF-101, hemos preentrenado el extractor de características que hemos utilizado después en el experimento principal. Durante esta fase de entrenamiento previa, hemos obtenido un modelo con una mejor capacidad de clasificación que el modelo basado en C3D (el extractor del trabajo original), con una mejora de más de 15 puntos porcentuales en la clasificación Top-1. Esta mejora se ha producido para todos los extractores de características propuestos, independientemente de la dimensión de la representación obtenida, lo cual pone de manifiesto que nos encontramos ante un extractor de mayor potencia.

Por otro lado, en el experimento final, que involucraba la detección de fotogramas anómalos dentro del conjunto de datos UCF-Crime, hemos observado cómo el uso del extractor de características con capas recurrentes obtiene unos resultados mejores que el sistema original. Nuestros modelos superaban a los dos modelos originales, tanto el preentrenado por los autores como la réplica entrenada por nosotros, en todas las métricas que hemos calculado. Podemos remarcar especialmente la mejora en la métrica AUC que hemos conseguido con el modelo de dimensión 768, ya que esta era la única métrica que se utilizaba en el artículo original para la comparación de modelos. Hemos conseguido una arquitectura que mejora a la propuesta inicial, por lo que consideramos que los experimentos propuestos han sido exitosos. Además, dado que en primera instancia consideramos que utilizar sólo esta métrica podía dar lugar a una comparación pobre, hemos utilizado otras métricas que dan información sobre el comportamiento del modelo en la clase positiva, obteniendo también resultados que superan a la experimentación original.

Otra mejora importante que hemos detectado es la capacidad de predicción de nuestros modelos a nivel de vídeo, en lugar de a nivel de fotogra-

ma. Aunque los resultados obtenidos por nuestros modelos no suponen una mejora tan representativa a la hora de localizar las anomalías dentro de los vídeos, sí que suponen un avance importante a la hora de detectar qué vídeos presentan anomalía. Concretamente, nuestro mejor modelo consigue una mejora de más de 10 puntos porcentuales sobre el modelo original en este contexto, lo cual es un aumento muy significativo.

Es importante destacar también que esta mejora en los resultados se ha producido a pesar de que nuestros extractores de características están entrenados, a priori, en un conjunto de datos de menor calidad que el extractor de características original. Mientras que el modelo convolucional 3D estaba entrenado en un conjunto de datos de más de 1000000 de vídeos y cerca de 500 clases, el nuestro está entrenado en un conjunto mucho más pequeño, de unos 10000 vídeos y 101 clases. Esta diferencia hace que el modelo original parta, presumiblemente, de una posición ventajosa respecto al nuestro, lo que hace que esta mejora resulte especialmente relevante.

Finalmente, a pesar de que los resultados obtenidos son mejores que los de la experimentación original, se puede observar que aún hay un amplio margen de mejora en este conjunto de datos. El número de falsos negativos es aún muy elevado, clasificándose correctamente menos del 50% de los fotogramas positivos. Es posible que esta problemática venga justificada, en parte, por el tipo de etiquetado del conjunto. Al tener que entrenar sin la localización exacta de las anomalías, resulta complicado enseñar al modelo a localizar de forma precisa la anomalía en el vídeo anómalo completo. Esto implica que, probablemente, se estén cometiendo errores en los primeros y últimos fotogramas alrededor de las anomalías. Además, hemos visto que en un cuarto de los vídeos etiquetados como anómalos no generamos ninguna etiqueta positiva, es decir, ignoramos casi el 25% de las anomalías presentes en el conjunto. Estamos hablando de un número muy importante de errores, que requerirán de modelos más potentes para ser detectados.

A raíz de las conclusiones obtenidas del estudio, exponemos a continuación posibles vías de trabajo futuro.

5.2. Trabajo futuro

Dados los problemas que hemos encontrado durante el desarrollo del trabajo, especialmente en el apartado práctico del mismo, aparecen las siguientes líneas de trabajo a investigar:

- Utilizar una base de datos de entrenamiento para el extractor de características de mayor tamaño: Por falta de capacidad de cómputo, no se han utilizado bases de datos de mayor tamaño para el entrenamiento del modelo que se usa posteriormente para la extracción de características. Probablemente, el uso de bases de datos con mayor diversidad producirá unos resultados mejores. El modelo original, como ya dijimos, está entrenado sobre Sports-1M, de tamaño significativamente mayor al empleado por nosotros. Existen conjuntos para clasificación de vídeos de mayor tamaño, como el conjunto YouTube-8M [74]. Puede ser interesante estudiar cómo el uso de un conjunto de datos u otro influye a la hora de entrenar el extractor de características. Teniendo en cuenta que los resultados obtenidos por nuestros modelos tras entrenar en el conjunto pequeño son comparables con los resultados originales, y mejores para la mayoría de las métricas calculadas, la mejora supuesta por un mejor preentrenamiento podría demostrar por completo que nos encontramos ante un modelo más potente.
- Afinar la arquitectura del modelo propuesto: En nuestra experimentación hemos propuesto un modelo basado en convoluciones 2D para extraer información de los fotogramas junto con una LSTM para extraer información temporal. En nuestra experimentación hemos estudiado el uso de tres representaciones de distintos tamaños, 512, 768 y 1024 elementos. No obstante, no se han explorado representaciones mayores, ya que los resultados obtenidos mejoraban la experimentación original y estamos ante modelos costosos, que requieren de muchas horas de cómputo para ser entrenados. Además, se ha utilizado Xception como red neuronal convolucional debido a su buen funcionamiento y pequeño tamaño, pero podríamos haber optado por otras arquitecturas disponibles. Es posible que las decisiones tomadas en el diseño hayan provocado que no nos encontremos ante el mejor modelo posible de este tipo y quede aún margen de mejora.
- Explorar nuevas arquitecturas para el extractor de características: Existen modelos llamados redes LSTM convolucionales [75] que sustituyen los productos internos de las LSTM clásicas por operaciones de convolución, por lo que son capaces de trabajar directamente con vídeos como dato de entrada. En este caso, no necesitaríamos una primera etapa de la red basada en una arquitectura convolucional, y podríamos aplicar directamente esta arquitectura. No obstante, tras primeras pruebas con este modelo, decidimos descartarlo por obtener malos resultados al ser entrenado completamente desde cero. Uniendo esta arquitectura

al uso de conjuntos de datos de mayor tamaño podrían mejorarse los resultados obtenidos.

- Modificar la política de entrenamiento del modelo: En nuestra experimentación hemos construido un modelo con una arquitectura similar al original, en el que hemos sustituido el extracto de características por uno que creímos de mayor potencia. No obstante, el resto del modelo se ha mantenido más o menos igual que el de partida para no influir de otra forma en el modelo. Debido a que el margen de mejora actual en el conjunto de datos es bastante grande, usar una política de entrenamiento distinta a la actual podría suponer una mejora en los resultados obtenidos, así que puede ser interesante explorar esta vía.
- Proponer modelos combinados: Los modelos que hemos utilizado en esta experimentación han sido estudiados de forma independiente, ya que nuestra intención era comprobar si las características espacio-temporales eran más potentes que las convolucionales puras para este problema. No hemos buscado, por tanto, obtener los mejores resultados posibles en el conjunto de datos. Durante la experimentación hemos observado cómo el modelo original y el modelo propuesto tienen características diferentes, y un buen comportamiento en distintos puntos (por ejemplo, para fotogramas fácilmente clasificables, el modelo original funciona ligeramente mejor que el nuestro). La utilización de los dos enfoques en un modelo combinado probablemente obtenga mejores resultados que los dos modelos por separado.

5.3. Publicaciones asociadas

Debido a los resultados obtenidos en el desarrollo del trabajo, tanto a nivel teórico como práctico, se han propuesto dos publicaciones relacionadas con el mismo. Dichas publicaciones son las siguientes:

- Luque-Sánchez, F., Hupont, I., Tabik, S., & Herrera, F. (2020). **Revisiting crowd behaviour analysis through deep learning: Taxonomy, anomaly detection, crowd emotions, datasets, opportunities and prospects.** *Information Fusion*. Esta publicación consiste en una revisión sobre el estado del arte en técnicas de análisis de multitudes en videovigilancia utilizando aprendizaje profundo. En dicho artículo se establece la taxonomía que se describe en el apartado teórico del trabajo, se revisan los principales trabajos que resuelven este problema utilizando

aprendizaje profundo, y se pone de manifiesto la necesidad de introducir características basadas en emociones para el análisis de multitudes.

- Luque-Sánchez, F., Hupont, I., Tabik, S., & Herrera, F. (2020). **Xception-LSTM: Deep Spatio-temporal features for crowd anomaly detection.** En preparación. Esta publicación extiende la experimentación de llevada a cabo en el trabajo a partir de las propuestas de trabajo futuro, para estudiar en profundidad los modelos espacio-temporales para la detección de anomalías. Se proponen nuevas arquitecturas basadas en capas CNN-LSTM combinadas, y se preentrenan los extractores de características en conjuntos de datos de mayor tamaño.

Bibliografía

- [1] Yunqian Ma y Gang Qian. *Intelligent video surveillance: systems and technology*. CRC Press, 2009.
- [2] M Sami Zitouni y col. “Advances and trends in visual crowd analysis: A systematic survey and evaluation of crowd modelling techniques”. En: *Neurocomputing* 186 (2016), págs. 139-159.
- [3] Waqas Sultani, Chen Chen y Mubarak Shah. “Real-world anomaly detection in surveillance videos”. En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, págs. 6479-6488.
- [4] HY Swathi, G Shivakumar y HS Mohana. “Crowd behavior analysis: A survey”. En: *2017 international conference on recent advances in electronics and communication technology (ICRAECT)*. IEEE. 2017, págs. 169-178.
- [5] Duc Thanh Nguyen, Wanqing Li y Philip O Ogunbona. “Human detection from images and videos: A survey”. En: *Pattern Recognition* 51 (2016), págs. 148-175.
- [6] Carolina Garate, Piotr Bilinsky y François Bremond. “Crowd event recognition using hog tracker”. En: *2009 Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*. IEEE. 2009, págs. 1-6.
- [7] Giuele Ciaparrone y col. “Deep learning in video multi-object tracking: A survey”. En: *Neurocomputing* 381 (2020), págs. 61-88.
- [8] Vijay Mahadevan y col. “Anomaly detection in crowded scenes”. En: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 2010, págs. 1975-1981.
- [9] Ramin Mehran, Alexis Oyama y Mubarak Shah. “Abnormal crowd behavior detection using social force model”. En: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, págs. 935-942.
- [10] Sergio A Velastin y Diego A Gómez-Lira. “People Detection and Pose Classification Inside a Moving Train Using Computer Vision”. En: *International Visual Informatics Conference*. Springer. 2017, págs. 319-330.

- [11] Cewu Lu, Jianping Shi y Jiaya Jia. "Abnormal event detection at 150 fps in matlab". En: *Proceedings of the IEEE international conference on computer vision*. 2013, págs. 2720-2727.
- [12] Wen Liu y col. "Future frame prediction for anomaly detection—a new baseline". En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, págs. 6536-6545.
- [13] Scott Blunsden y RB Fisher. "The BEHAVE video dataset: ground truthed video for multi-person behavior classification". En: *Annals of the BMVA* 4.1-12 (2010), pág. 4.
- [14] M. S. Ryoo y J. K. Aggarwal. *UT-Interaction Dataset, ICPRI contest on Semantic Description of Human Activities (SDHA)*. 2010. URL: http://cvrc.ece.utexas.edu/SDHA2010/Human%5C_Interaction.html.
- [15] Enrique Bermejo Nievas y col. "Movies Fight Detection Dataset". En: *Computer Analysis of Images and Patterns*. Springer. 2011, págs. 332-339. URL: <http://visilab.etsii.uclm.es/personas/oscar/FightDetection/>.
- [16] Enrique Bermejo Nievas y col. "Hockey Fight Detection Dataset". En: *Computer Analysis of Images and Patterns*. Springer. 2011, págs. 332-339. URL: <http://visilab.etsii.uclm.es/personas/oscar/FightDetection/>.
- [17] Bernhard Schölkopf y col. "Support vector method for novelty detection". En: *Advances in neural information processing systems*. 2000, págs. 582-588.
- [18] Dan Xu y col. "Learning deep representations of appearance and motion for anomalous event detection". En: *arXiv preprint arXiv:1510.01553* (2015).
- [19] Pascal Vincent y col. "Extracting and composing robust features with denoising autoencoders". En: *Proceedings of the 25th international conference on Machine learning*. 2008, págs. 1096-1103.
- [20] Berthold KP Horn y Brian G Schunck. "Determining optical flow". En: *Techniques and Applications of Image Understanding*. Vol. 281. International Society for Optics y Photonics. 1981, págs. 319-331.
- [21] Matheus Gutoski y col. "Detection of video anomalies using convolutional autoencoders and one-class support vector machines". En: *XIII Brazilian Congress on Computational Intelligence*. Vol. 2017. 2017.
- [22] John Canny. "A computational approach to edge detection". En: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), págs. 679-698.
- [23] Meng Yang y col. "Deep Learning and One-class SVM based Anomalous Crowd Detection". En: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2019, págs. 1-8.

- [24] Bruce D Lucas, Takeo Kanade y col. “An iterative image registration technique with an application to stereo vision”. En: (1981).
- [25] Geoffrey E Hinton, Simon Osindero y Yee-Whye Teh. “A fast learning algorithm for deep belief nets”. En: *Neural computation* 18.7 (2006), págs. 1527-1554.
- [26] Zhijun Fang y col. “Abnormal event detection in crowded scenes based on deep learning”. En: *Multimedia Tools and Applications* 75.22 (2016), págs. 14617-14639.
- [27] Yuming Fang y col. “Bottom-up saliency detection model based on human visual sensitivity and amplitude spectrum”. En: *IEEE Transactions on Multimedia* 14.1 (2011), págs. 187-198.
- [28] Tsung-Han Chan y col. “PCANet: A simple deep learning baseline for image classification?” En: *IEEE transactions on image processing* 24.12 (2015), págs. 5017-5032.
- [29] Sorina Smeureanu y col. “Deep appearance features for abnormal behavior detection in video”. En: *International Conference on Image Analysis and Processing*. Springer. 2017, págs. 779-789.
- [30] Ken Chatfield y col. “Return of the devil in the details: Delving deep into convolutional nets”. En: *arXiv preprint arXiv:1405.3531* (2014).
- [31] Jiayu Sun, Jie Shao y Chengkun He. “Abnormal event detection for video surveillance using deep one-class learning”. En: *Multimedia Tools and Applications* 78.3 (2019), págs. 3633-3647.
- [32] Kuldeep Singh y col. “Crowd anomaly detection using aggregation of ensembles of fine-tuned ConvNets”. En: *Neurocomputing* 371 (2020), págs. 188-198.
- [33] Shaonian Huang, Dongjun Huang y Xinmin Zhou. “Learning multi-modal deep representations for crowd anomaly event detection”. En: *Mathematical Problems in Engineering* 2018 (2018).
- [34] Honglak Lee y col. “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations”. En: *Proceedings of the 26th annual international conference on machine learning*. 2009, págs. 609-616.
- [35] Ryota Hinami, Tao Mei y Shin'ichi Satoh. “Joint detection and recouning of abnormal events by learning deep generic knowledge”. En: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, págs. 3619-3627.

- [36] Ross Girshick. "Fast r-cnn". En: *Proceedings of the IEEE international conference on computer vision*. 2015, págs. 1440-1448.
- [37] Mohammad Sabokrou y col. "Real-time anomaly detection and localization in crowded scenes". En: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2015, págs. 56-62.
- [38] Mohammad Sabokrou y col. "Fast and accurate detection and localization of abnormal behavior in crowded scenes". En: *Machine Vision and Applications* 28.8 (2017), págs. 965-985.
- [39] Mohammad Sabokrou y col. "Deep-cascade: Cascading 3d deep neural networks for fast anomaly detection and localization in crowded scenes". En: *IEEE Transactions on Image Processing* 26.4 (2017), págs. 1992-2004.
- [40] Yachuang Feng, Yuan Yuan y Xiaoqiang Lu. "Learning deep event models for crowd anomaly detection". En: *Neurocomputing* 219 (2017), págs. 548-556.
- [41] Cinzia Viroli y Geoffrey J McLachlan. "Deep gaussian mixture models". En: *Statistics and Computing* 29.1 (2019), págs. 43-51.
- [42] Anitha Ramchandran y Arun Kumar Sangaiah. "Unsupervised deep learning system for local anomaly event detection in crowded scenes". En: *Multimedia Tools and Applications* (2019), págs. 1-21.
- [43] Mahdyar Ravanbakhsh y col. "Plug-and-play cnn for crowd motion analysis: An application in abnormal event detection". En: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2018, págs. 1689-1698.
- [44] Shifu Zhou y col. "Spatial-temporal convolutional neural networks for anomaly detection and localization in crowded scenes". En: *Signal Processing: Image Communication* 47 (2016), págs. 358-368.
- [45] Mahdyar Ravanbakhsh y col. "Training adversarial discriminators for cross-channel abnormal event detection in crowds". En: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2019, págs. 1896-1904.
- [46] Ian Goodfellow y col. "Generative adversarial nets". En: *Advances in neural information processing systems*. 2014, págs. 2672-2680.
- [47] Krishan Kumar, Anurag Kumar y Ayush Bahuguna. "D-CAD: Deep and crowded anomaly detection". En: *Proceedings of the 7th International Conference on Computer and Communication Technology*. 2017, págs. 100-105.

- [48] Laurenz Wiskott y Terrence J Sejnowski. "Slow feature analysis: Unsupervised learning of invariances". En: *Neural computation* 14.4 (2002), págs. 715-770.
- [49] Mohammad Sabokrou y col. "Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes". En: *Computer Vision and Image Understanding* 172 (2018), págs. 88-97.
- [50] Jun Wang y Limin Xia. "Abnormal behavior detection in videos using deep learning". En: *Cluster Computing* 22.4 (2019), págs. 9229-9239.
- [51] Nian Chi Tay y col. "A robust abnormal behavior detection method using convolutional neural network". En: *Computational Science and Technology*. Springer, 2019, págs. 37-47.
- [52] AS Keçeli y AYDIN Kaya. "Violent activity detection with transfer learning method". En: *Electronics Letters* 53.15 (2017), págs. 1047-1048.
- [53] Igor Kononenko, Edvard Šimec y Marko Robnik-Šikonja. "Overcoming the myopia of inductive learning algorithms with RELIEFF". En: *Applied Intelligence* 7.1 (1997), págs. 39-55.
- [54] Swathikiran Sudhakaran y Oswald Lanz. "Learning to detect violent videos using convolutional long short-term memory". En: *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE. 2017, págs. 1-6.
- [55] Mark Marsden y col. "ResnetCrowd: A residual deep learning architecture for crowd counting, violent behaviour detection and crowd density level classification". En: *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE. 2017, págs. 1-7.
- [56] Wei Song y col. "A novel violent video detection scheme based on modified 3D convolutional neural networks". En: *IEEE Access* 7 (2019), págs. 39172-39179.
- [57] E Fenil y col. "Real time violence detection framework for football stadium comprising of big data analysis and deep learning through bidirectional LSTM". En: *Computer Networks* 151 (2019), págs. 191-200.
- [58] Shakil Ahmed Sumon y col. "Violent crowd flow detection using deep learning". En: *Asian Conference on Intelligent Information and Database Systems*. Springer. 2019, págs. 613-625.

- [59] Guoan Cheng y col. "Abnormal behavior detection for harbour operator safety under complex video surveillance scenes". En: *2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*. IEEE. 2017, págs. 324-328.
- [60] Wei Liu y col. "Ssd: Single shot multibox detector". En: *European conference on computer vision*. Springer. 2016, págs. 21-37.
- [61] Du Tran y col. "Learning spatiotemporal features with 3D convolutional networks". En: *Proceedings of the IEEE international conference on computer vision*. 2015, págs. 4489-4497.
- [62] Andrej Karpathy y col. "Large-scale video classification with convolutional neural networks". En: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2014, págs. 1725-1732.
- [63] Sepp Hochreiter y Jürgen Schmidhuber. "Long short-term memory". En: *Neural computation* 9.8 (1997), págs. 1735-1780.
- [64] François Chollet. "Xception: Deep learning with depthwise separable convolutions". En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, págs. 1251-1258.
- [65] Jia Deng y col. "Imagenet: A large-scale hierarchical image database". En: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, págs. 248-255.
- [66] François Chollet y col. *Keras*. <https://keras.io>. 2015.
- [67] Khurram Soomro, Amir Roshan Zamir y Mubarak Shah. "UCF101: A dataset of 101 human actions classes from videos in the wild". En: *arXiv preprint arXiv:1212.0402* (2012).
- [68] Gary Bradski y Adrian Kaehler. "OpenCV". En: *Dr. Dobb's journal of software tools* 3 (2000).
- [69] Travis E Oliphant. *A guide to NumPy*. Vol. 1. Trelgol Publishing USA, 2006.
- [70] Stefan Van Der Walt, S Chris Colbert y Gael Varoquaux. "The NumPy array: a structure for efficient numerical computation". En: *Computing in Science & Engineering* 13.2 (2011), pág. 22.
- [71] The pandas development team. *pandas-dev/pandas: Pandas*. Ver. latest. Feb. de 2020. DOI: [10.5281/zenodo.3509134](https://doi.org/10.5281/zenodo.3509134). URL: <https://doi.org/10.5281/zenodo.3509134>.
- [72] Martín Abadi y col. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: <https://www.tensorflow.org/>.

- [73] Francisco Luque Sánchez y col. “Revisiting crowd behaviour analysis through deep learning: Taxonomy, anomaly detection, crowd emotions, datasets, opportunities and prospects”. En: *Information Fusion* (2020).
- [74] Sami Abu-El-Haija y col. “Youtube-8m: A large-scale video classification benchmark”. En: *arXiv preprint arXiv:1609.08675* (2016).
- [75] SHI Xingjian y col. “Convolutional LSTM network: A machine learning approach for precipitation nowcasting”. En: *Advances in neural information processing systems*. 2015, págs. 802-810.