PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

ESCUELA DE POSGRADO

IDENTIFICACIÓN AUTOMÁTICA DE ACCIONES HUMANAS EN SECUENCIAS DE VIDEO PARA SOPORTE DE VIDEOVIGILANCIA

TESIS PARA OPTAR EL GRADO ACADÉMICO DE MAGÍSTER EN

INFORMÁTICA CON MENCIÓN EN CIENCIAS DE LA COMPUTACIÓN

AUTOR

LUIS CHRISTIAN FERNÁNDEZ MARTÍNEZ

ASESOR

DR. CÉSAR ARMANDO BELTRÁN CASTAÑÓN

LIMA – PERÚ 2017

Dedicatoria

A mi amada Kelly y a mis pequeños amores, Kristen y Kayden.

Agradecimientos

A Dios, por sus bendiciones.
Al Doctor César Beltrán por su orientación y paciencia.
Al Grupo Avatar por facilitarme las instalaciones del laboratorio y el equipo para poder ejecutar los algoritmos.
A Enrique Chiroque, Andrea Cáceres, Axel Muñoz, Vanessa Vega y a todo el equipo humano del Grupo Avatar, por la paciencia y apoyo.
A mis padres por su aliento y empuje.

Tabla de contenidos

1. Resumen

La identificación de acciones en secuencias de video es un tema de especial interés para aplicaciones como detección de peleas, identificación de vandalismo, detección de asaltos a transeúntes, detección de contenido no apto para menores, etc. Este interés se encuentra asociado al incremento de cámaras de videovigilancia alrededor del mundo y a la masiva producción de videos en línea cargados a las diferentes plataformas sociales de almacenamiento y distribución de contenido bajo demanda. Debido a ello, se decide utilizar un modelo de detección de acciones humanas y aplicarlo en secuencias de videovigilancia. Dicho modelo utiliza redes neuronales profundas, con la finalidad de poder realizar la tarea de clasificación. El modelo aplicado se basa en el extracción de características convolucionales y temporales utilizando una parte de la red Inception V3 para lo primero y una red LSTM para lo segundo. Finalmente, se aplica el modelo en el dataset UCF101 el cual contiene acciones humanas diversas y luego sobre el dataset VIRAT 2.0 Ground, el cual contiene secuencias de videovigilancia.

2. Generalidades

2.1. Descripción

La detección de violencia en video es un tópico de estudio que ha adquirido un interés especial en los últimos años, esta tendencia está asociada al incremento de cámaras de videovigilancia alrededor del mundo y a la masiva producción de videos en línea cargados a las diferentes plataformas sociales de almacenamiento y distribución de contenido bajo demanda (De Souza, Chavez, do Valle Jr, Eduardo A, & Araújo, 2010; Nievas, Suarez, García, & Sukthankar, 2011). Dentro de este escenario, las aplicaciones de identificación de violencia son diversas, como por ejemplo, detección de peleas en cárceles, reclusorios o escuelas, identificación de vandalismo en zonas urbanas, detección de asaltos a transeúntes (Rota, Conci, Sebe, & Rehg, 2015), además de, identificación automática de contenido no apto para menores en grandes cantidades de videos cargados a las plataformas sociales o transmitido en tiempo real, (Deniz, Serrano, Bueno, & Kim, 2014). Adicionalmente, el impacto de la automatización de este proceso se traduce en términos de mejora del tiempo de respuesta ante situaciones concretas de violencia en video, considerando que, los seres humanos reducen su capacidad y efectividad al realizar la tarea de identificación, durante largas jornadas, analizando grandes cantidades de video de diferentes fuentes (Deniz et al., 2014).

Las técnicas propuestas para la detección de actividad humana en video son diversas, debido a ello, para poder clasificarlas, Aggarwal y Ryoo proponen una taxonomía basada en la necesidad de identificar sub eventos o acciones atómicas a fin de construir de forma jerárquica la acción principal, o de definir la acción de forma directa a partir de los datos de entrada (Aggarwal & Ryoo, 2011). Por un lado, a las técnicas de identificación de acciones de forma directa se denominan enfoques de una capa, éstos están basados en el procesamiento de los frames o secuencias de imágenes del video: análisis espacio-temporal volumétrico, de trayectorias o de características, o análisis secuencial basado en ejemplares o poses, o de estados de modelos simplificados de la estructura humana (Aggarwal & Ryoo, 2011). Por otro lado, los métodos de identificación de acciones a partir de sub eventos se denominan enfoques jerárquicos, los cuales describen las actividades humanas como un conjunto de

acciones más simples, utilizando tres tipos de metodologías: i) de reconocimiento estadístico —basado modelos probabilísticos—, ii) sintáctico —utilizando sintaxis gramaticales— o iii) descriptivo —utilizando estructuras espaciales, temporales y lógicas a partir de las acciones simples que componen una actividad humana. (Aggarwal & Ryoo, 2011)

Por tanto, el problema que se abordó en este trabajo de tesis consistió en ¿cómo identificar acciones humanas en secuencias de videovigilancia con la finalidad de automatizar el monitoreo de videos obtenidos de cámaras de seguridad?. Esta investigación adquiere importancia debido a que contribuye a enriquecer la aplicabilidad del tópico en cuestión, el cual está enmarcado en el ámbito de la visión computacional y el diseño y aplicación de redes neuronales profundas.

Además se debe considerar que, para diseñar una solución a la problemática de la detección de violencia en video —entendiéndose la violencia como un tipo de actividad humana—, se debe acotar qué se considerará como violencia. Por ello, se utilizará como base la definición de la Organización Mundial de la Salud que describe la violencia como: “el uso intencional de la fuerza física, amenazas contra uno mismo, otra persona, un grupo o una comunidad que tiene como consecuencia o es muy probable que tenga como consecuencia un traumatismo, daños psicológicos, problemas de desarrollo o la muerte” (Organización Mundial de la Salud, 2012). En consecuencia, a partir de esta definición, se identifican algunos volúmenes de datos de video que posean un conjunto acotado de acciones humanas que permitan comparar el desempeño del aprendizaje profundo en la tarea de identificación de acciones. Adicionalmente, la diversidad de aspectos no controlados en las escenas, así como el ruido, la resolución del video, distancia de la cámara, entre otros, serán procesados de forma conjunta por el algoritmo.

2.2. Objetivo General

Identificar acciones humanas en secuencias de videovigilancia usando técnicas de aprendizaje profundo con la finalidad de apoyar a los sistemas de monitoreo por video.

2.3. Objetivos Específicos

Los objetivos específicos son:

OE1. Identificar al menos dos conjuntos de datos de video con secuencias de acciones humanas, en ambientes no controlados, para entrenar los modelos neuronales profundos.
OE2. Implementar al menos una arquitectura de red neuronal profunda para la identificación de acciones humanas en secuencias de video.
OE3. Comparar las métricas de desempeño de un modelo de red en los conjuntos de datos de video seleccionados.

2.4. Resultados Esperados

RE1. Carpeta que contenga las muestras de cada conjunto de datos, distribuida en subconjuntos de entrenamiento y validación, además de subdividida según la clase de acción humana, y preprocesada para su utilización con los modelos neuronales seleccionados.
RE2. Implementación del modelo neuronal de forma que se pueda evaluar dentro de las mismas condiciones de hardware y software.
RE3. Resultados comparativos del desempeño del modelo neuronal en cada uno de los conjuntos de datos.

2.5. Alcance

El presente proyecto de investigación abarca la descripción de conceptos y búsqueda del estado del arte de investigaciones centradas en la identificación y clasificación de acciones humanas en video. Las técnicas que se describen utilizan métodos manuales de extracción de características y métodos automatizados que utilizan redes neuronales para la extracción y clasificación.

Para la identificación de acciones se utilizará una arquitectura de dos fases, la primera es una red convolucional para la extracción de características de cada frame y la segunda, un red LSTM para la captura de las características temporales y su posterior clasificación.

Esta arquitectura será aplicada a dos bases de datos: UCF-101 y Virat 2.0. Se utilizará

dos versiones del dataset Virat 2.0, de procesamiento de frame completo y de frame recortado. Finalmente, los resultados serán comparados en las secciones finales del documento.

2.6. Metodología De Desarrollo

Para lograr el objetivo planteado se debe identificar un modelo de red convolucional para la clasificación de video, luego se debe implementar dicho modelo, además, se debe identificar al menos dos bases de datos de entrenamiento y, finalmente, se debe evaluar dicho modelo de clasificación de acciones en video.

El primer lugar, se implementará un modelo de red neuronal basado en el estado del arte encontrado hasta el mes de mayo de 2017, con la finalidad de explorar algunas experiencias previas en el proceso de categorización de acciones humanas en secuencias de frames. Como resultado de dicha exploración se obtendrá la implementación de una arquitectura de red neuronal profunda para la clasificación de acciones humanas en video.

Luego de ello, se pre-procesará al menos dos conjuntos de datos de acciones humanas en video. Dichos videos deben haber sido capturados en ambientes no controlados, esto con la finalidad de obtener un algoritmo capaz de clasificar acciones tomadas en espacios videovigilados.

Finalmente, se realizará un entrenamiento de la red neuronal utilizando los conjuntos de datos propuestos. Las métricas y resultados de dicho entrenamiento serán detalladas en las siguientes secciones del documento.

Todo este proceso metodológico planteado para alcanzar los objetivos de esta investigación se aprecia en la Figura 1:

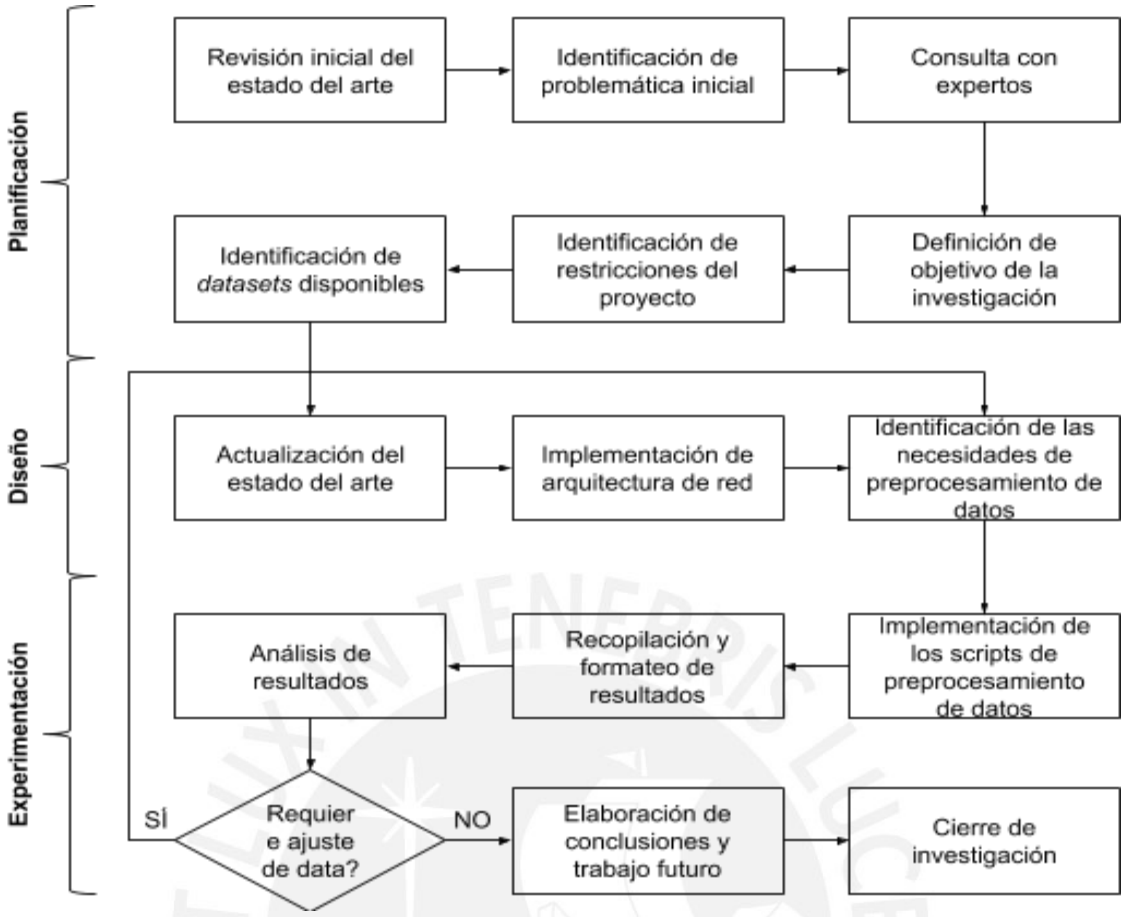

Figura 1: Metodología utilizada en el proyecto de investigación

Fase de planificación: Identificación en el estado del arte de un modelo de red neuronal profunda para identificar y clasificar acciones humanas en secuencias de frames. Dicho modelo debe poder ejecutarse en un GPU con al menos 700 CUDA cores, con la finalidad de poder procesarlo en alguno de los computadores disponibles en los laboratorios de la universidad. Además de la identificación de dos *dataset* de acciones humanas.

Fase de diseño: Programar el modelo de red neuronal seleccionado de forma que pueda procesar las bases de datos de secuencias de videos de acciones humanas. Pre-procesar los conjuntos de datos de video, agrupándolos por categorías, redimensionando los frames y modificando la cantidad de frames de cada muestra en el dataset con la finalidad de poder procesarlos en la red neuronal seleccionada.

Fase de experimentación: Entrenar la red neuronal con cada conjunto de datos tomando las métricas obtenidas para definir la idoneidad del modelo seleccionado para la aplicación en el dominio de acciones en video.

3. Marco Conceptual

Este capítulo describe los conceptos generales de las redes neuronales, de ellas, las redes convolucionales y las recurrentes, las cuales son usadas en el procesamiento de video y de secuencias de imágenes.

3.1. Redes neuronales artificiales

Las redes neuronales artificiales son un paradigma de programación de inspiración biológica que permite a un algoritmo aprender a partir de un conjunto de datos observacionales (Ali & Senan, 2016). Tal como se aprecia en la Figura 2, una red de esta naturaleza está compuesta por neuronas que procesan la suma ponderada de las señales de entrada y generan una salida que oscila entre 0 y 1, de forma discreta tal como lo proponen originalmente McCulloch y Pitts (McCulloch & Pitts, 1943) o de forma contínua si se utiliza otro tipo de función de activación (Anil K. Jain, Mao, & Mohiuddin, 1996). El aprendizaje en las redes artificiales se produce al obtener una medida de la señal del error obtenido, comparando las predicciones de la red y los valores esperados, con la finalidad de ajustar los pesos W_i, de forma iterativa, para gradualmente reducir dicha medida del error (Jain et al., 1996).

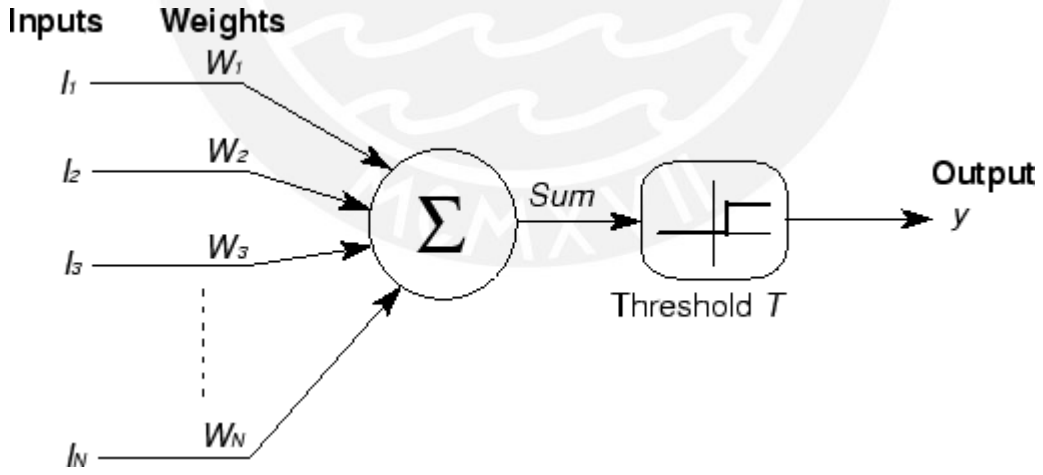

Figura 2: Diagrama de una neurona con una función de activación de umbral binario (Kawaguchi, 2000)

3.2. Algoritmo de propagación hacia atrás (*backpropagation*)

Para comprender el proceso de propagación hacia atrás, se deben definir dos conceptos previamente: la propagación hacia adelante y la función de costo. En primer lugar, se denomina propagación hacia adelante o *forward-propagation* al cómputo que se realiza sobre una entrada *x* de una red neuronal con la finalidad de producir una salida *ŷ* la cual es la predicción que realiza el modelo de la clase *y* a la que pertenece *x* tal como se aprecia en la Figura 2 donde el vector de entrada *x* posee $I_{1..N}$ componentes. En segundo lugar, se tiene un valor escalar denominado función de costo o función de pérdida $J(\theta)$, donde θ es el vector de parámetros de la red neuronal que busca minimizar el valor de la función J que representa la diferencia entre el valor estimado *ŷ* y el valor real *y*. Como consecuencia, al proceso de propagar hacia atrás la información de la función de costo, con la finalidad de computar la gradiente y poder minimizar dicha función, se denomina propagación hacia atrás (Goodfellow, Bengio, & Courville, 2016). Tal como se aprecia en la Figura 3, donde la operación asociada al nodo en la propagación hacia adelante se representa por la función f_i y la derivada de la función en la propagación hacia atrás se representa por la función f'_i(Rojas, 1996).

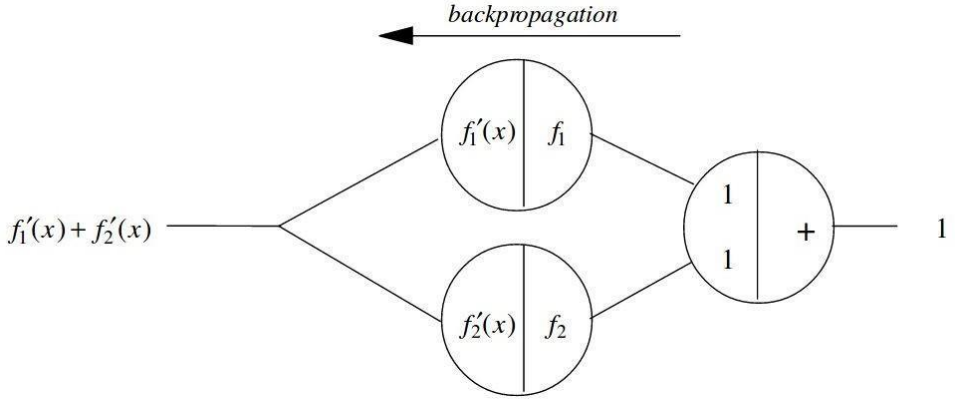

Figura 3: Propagación hacia atrás en una red aditiva de dos funciones primitivas (Rojas, 1996)

El algoritmo de retropropagación o *backpropagation* en una red artificial multicapa es el siguiente (Jain et al., 1996): (i) los pesos iniciales en cada unidad o neurona de la red son inicializados con pequeños valores aleatorios; (ii) para luego procesar un patrón de entrada que produce una señal que se propaga por toda la red hasta la última capa; (iii) después se calcula la salida o predicción de la red; (iv) y se computa el delta para cada capa precedente propagando el error en sentido inverso desde la última capa hasta la primera; (v) luego de ello, se actualizan los pesos en toda la red; (vi)

finalmente se repite el proceso desde el paso (ii).

3.3. Redes Neuronales Convolucionales (CNNs)

Las Redes Neuronales Convolucionales (CNNs, por sus siglas en inglés) son un tipo especial de redes neuronales, cuyo desarrollo fue motivado por la investigación neurobiológica de células nerviosas selectivas y localmente sensibles en la corteza visual (Ali & Senan, 2016). La operación que determina su nombre es la convolución, la cual se entiende como la integral que expresa la cantidad de solapamiento de una función *g,* a medida que se desplaza sobre otra función *f*, es decir, "combina" una función con otra (Weisstein, 2002).

$$f(t) * g(t) = \int_0^t f(\tau) g(t - \tau) d\tau, \qquad (1)$$

Es así que las redes convolucionales se entienden como un tipo especial de redes neuronales de alimentación directa que se conforman de varias capas y que pueden extraer propiedades topológicas de una imagen de forma automatizada. En donde, de la misma manera que la mayoría de redes neuronales, es entrenada con una versión del algoritmo de propagación hacia atrás. Además, este tipo de redes neuronales están diseñadas para reconocer patrones visuales directamente de imágenes digitales con poco o ningún preprocesamiento. Debido a ello, las redes convolucionales son capaces de extraer patrones con extrema variabilidad, como por ejemplo, texto manuscrito e imágenes naturales. Finalmente, la arquitectura típica de una CNN está conformada por un conjunto de capas de convolución, capas de reducción o de submuestreo y una o más capas completamente conectada, tal como se muestra en la Figura 4 —donde una muestra de una imagen de una señal de tránsito es procesada en una red neuronal con dos capas convolucionales, dos capas de submuestreo y una capa completamente conectada. En dicha estructura, las capas de convolución y submuestreo se diseñan de forma alternada (A. K. Jain, Jianchang Mao, & Mohiuddin, 1996).

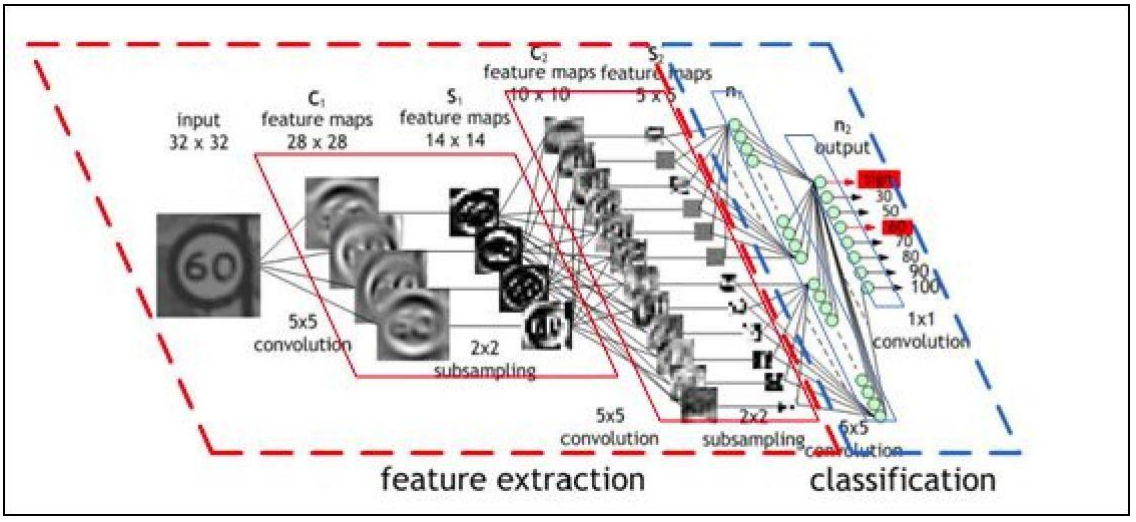

Figura 4: Proceso de extracción de características, usando capas convolucionales y reductoras, y de clasificación al final de la red multicapa (Peemen, 2013).

3.4. Redes Neuronales Recurrentes (RNN)

Las redes neuronales recurrentes (RNN) son un tipo especial de redes neuronales que se caracterizan por poseer conecciones internas que apuntan a sí mismas. Es decir, mientras que en las redes neuronales convolucionales se utiliza la retropropagación para ajustar los pesos de la red y así lograr el aprendizaje, en las Redes neuronales recurrentes (RNN) el aprendizaje está modelado por neuronas con conexiones recurrentes las cuales procesan cada nuevo patrón para obtener una salida que permite modificar la entrada de cada neurona cambiando el estado de toda la red (Jain et al., 1996). Esta naturaleza le permite a la red comprimir toda la memoria en un espacio de baja dimensión. Esto conlleva a desarrollar la característica más importante de las redes recurrentes: formar memoria a corto plazo (Mikolov, Karafit, Burget, Cernock, & Khudanpur, 2010).

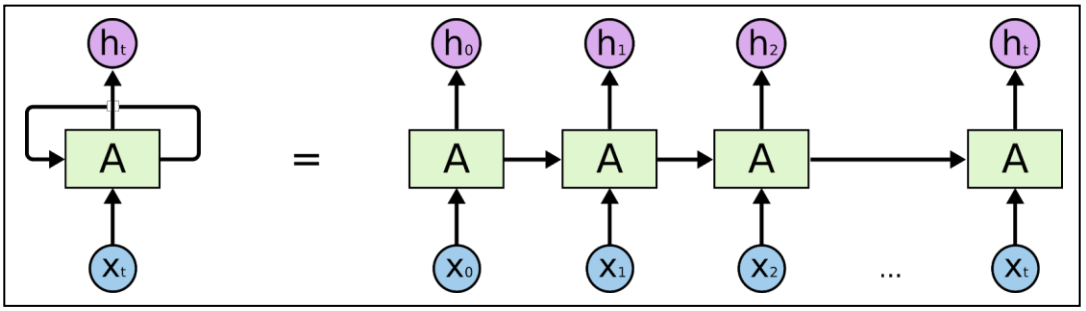

Figura 5: Red neuronal recurrente “desenrollada” (Chen & Xia, 2015)

3.5. Long Short-Term Memory (LSTM)

Las redes recurrentes utilizan las conexiones de retroalimentación para almacenar representaciones de eventos de entrada recientes en forma de funciones de activación, es decir poseen "memoria a corto plazo". Dicha característica es la más importante en las RNN, sin embargo no siempre resulta ser adecuada para el procesamiento de secuencias de datos, especialmente cuando el espacio entre las señales maestras es muy grande (Hochreiter & Schmidhuber, 1997). Debido a ello Hochreiter propuso una arquitectura basada en bloques de memoria recurrentes que permite almacenar señales entre intervalos de tiempo de incluso más de 1000 pasos, sin pérdida de la capacidad original para intervalos pequeños. Esto se logra mediante un algoritmo eficiente basado en gradiente para una arquitectura de flujo de error constante a través de estados internos de los bloques truncando la gradiente en ciertos puntos específicos de la arquitectura, es decir, almacenando selectivamente la información. Como se puede apreciar en la Figura 6, las entradas h_{t-1} y x_t se concatenan e introducen en cuatro unidades con funciones σ y *tanh,* cuyas salidas se conmutan en una línea transversal que almacena siempre el estado de la celda anterior y se propaga a lo largo de toda la red. Finalmente, cada celda, produce una salida h_t que contiene la salida de cada celda la cual, al mismo tiempo, se propaga hacia la siguiente para su posterior procesamiento. (Chen & Xia, 2015).

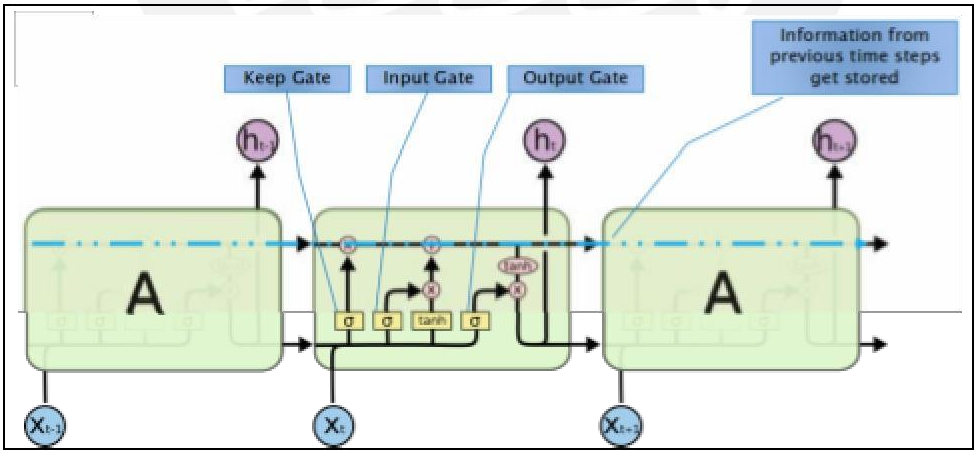

Figura 6: Funcionamiento de las puertas de control de memoria en un bloque LSTM (Chen & Xia, 2015)

4. Revisión del Estado del Arte

Para la revisión del estado del arte se identificaron las bases de datos Springer, IEEE y ACM. Aunque, también fue necesario realizar búsquedas apoyándose en Google Scholar. Para ello, se realizaron tres importantes grupos de búsqueda. En la primera búsqueda se utilizaron los términos: “violence detection”, “video surveillance”. En la segunda: “action recognition”, “human action recognition”, “video clasification”, “action clasification”. Y en la última, se añadieron los términos “deep learning” y “neural networks” a los términos anteriores. Adicionalmente, las búsquedas arrojaron algunos artículos con revisiones sistemáticas del estado del arte. Estas revisiones también fueron consideradas y se incluyó parte de sus referencias en la bibliografía.

4.1. Taxonomía de técnicas de reconocimiento de acciones humanas

El proceso de reconocimiento de violencia en secuencias de video se encuentra enmarcado en el área del reconocimiento de actividades humanas. Dicha área de investigación —reconocimiento de acciones— viene desarrollándose desde la década de los setenta de la mano de autores como Johansson’s y Aggarwal (Aggarwal & Ryoo, 2011).

Dentro de dicho campo de estudio, se han utilizado diferentes técnicas de reconocimiento de acciones las cuales pueden ser clasificadas, según la taxonomía de Aggarwal (Aggarwal & Ryoo, 2011), en dos grandes grupos: directo o indirecto, dependiendo del método de identificación que se utilice. Tal como se muestra en la Figura 7, el primer conjunto de enfoques de la rama izquierda realiza la categorización directamente a partir de la secuencia de imágenes de entrada, mientras que el segundo,de la rama derecha, estructura las actividades analizadas a partir de un conjunto atómico de subeventos o estados para finalmente interpretar la estructura generada de forma indirecta. Además, según esta taxonomía, dentro del primer gran grupo de métodos de identificación de acciones directas, existe una subclasificación importante: métodos espacio-temporales y métodos secuenciales. Por otro lado, dentro del segundo grupo de métodos de identificación, es decir, los enfoques jerárquicos, existe también una subdivisión adicional: métodos indirectos estadísticos, indirectos sintácticos e indirectos descriptivos.

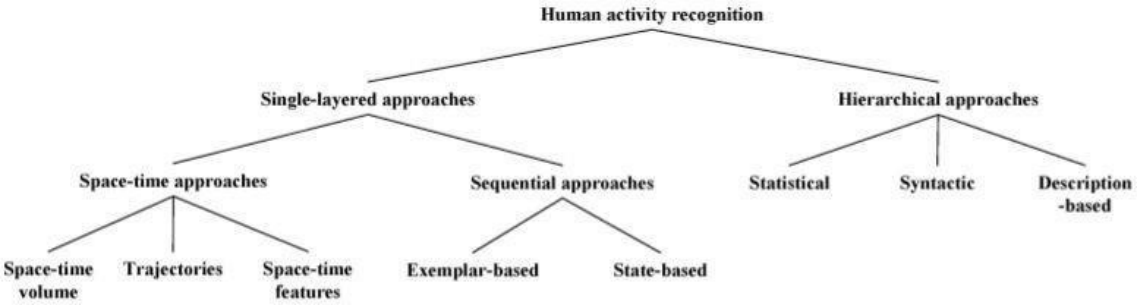

Figura 7: Taxonomía de técnicas de reconocimiento de acciones humanas (Aggarwal & Ryoo, 2011)

4.1.1. Enfoques espacio-temporales

Por una parte, los enfoques espacio-temporales se subdividen en tres tipos: volumétricos, de trayectorias y de características. En donde los primeros clasifican la actividad humana utilizando la dimensión del tiempo para construir modelos volumétricos espacio-temporales como el utilizado por Hu *et al.* (Hu et al., 2009); quien propone combinar la información acumulada del movimiento de la secuencia de frames o *Motion History Image* (MHI) con la información aparente obtenida al sustraer el fondo del frame. Por otra parte, los enfoques basados en trayectorias construyen los modelos a partir del trayecto que describen los nodos o puntos de interés identificados en las figuras humanas de cada *frame* tal como plantea Wang *el al.* (Wang, Klser, Schmid, & Liu, 2011) quien muestrea cada cuadro del video utilizando puntos densos para luego modelar las acciones contenidas en el video mediante trayectorias densas utilizando como base la información de desplazamiento de campo de flujo óptico denso. Finalmente, los enfoques de características espacio-temporales procesan cada frame para construir un volumen tridimensional de características, tal como lo hace Yu *et al.* (Yu, Kim, & Cipolla, 2010) que utiliza la combinación de un clasificador *K-means Forest* con un *Random Forest* aplicados a una estructura de información de acciones humanas y una bolsa de *textones* semánticos[1] generados a partir de un conjunto de bosques semánticos de *textones* (SFTs por sus siglas en inglés). Tal como se muestra en la Figura 8, estos SFTs son construidos a partir de volúmenes espacio-temporales de puntos de interés localizados utilizando un detector V-FAST para lograr el reconocimiento de acciones en tiempo real.

[1] Un textón semántico, también denominado *visual word,* es el resultado de un conjunto de filtros aplicados a una imagen a fin de obtener una representación discreta o conjunto de características de bajo de nivel de la misma (Shotton, Johnson, & Cipolla, 2008).

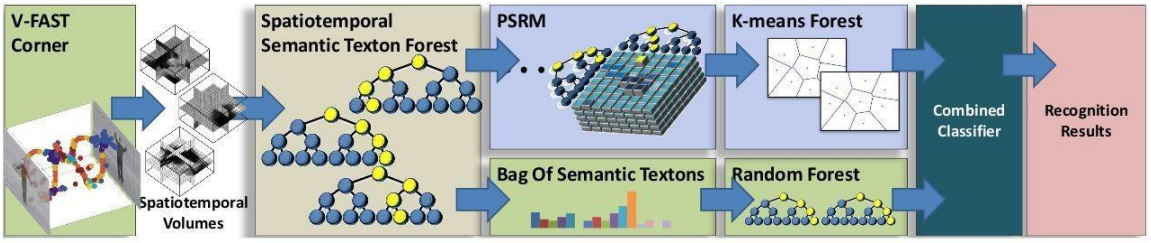

Figura 8: Enfoque propuesto por Yu et al. para la identificación de acciones utilizando volúmenes espacio-temporales de características (Yu et al., 2010).

4.1.2. Métodos de identificación secuenciales

Por otra parte, los métodos de identificación secuenciales convierten las secuencias de imágenes del video en una secuencia de vectores de características con los cuales se construyen plantillas contra las que serán comparadas las nuevas secuencias a clasificar. Por último, los métodos secuenciales utilizan modelos de estados simplificados de la estructura humana para poder realizar la identificación de la acción (Cheng, Wan, Saudagar, Namuduri, & Buckles, 2015).

4.1.3. Enfoques jerárquicos

Finalmente, los enfoques jerárquicos de detección descomponen las actividades humanas en un conjunto de acciones más simples con la finalidad de modelar la compleja estructura de dichas actividades (Cheng et al., 2015). Ésta a su vez se subdivide en metodologías de reconocimiento del tipo estadístico —usando modelos probabilísticos—, sintáctico —usando sintaxis gramaticales— o descriptivo —utilizando estructuras espaciales, temporales y lógicas a partir de las acciones simples que componen una actividad humana.

4.2. Reconocimiento de acciones humanas usando redes neuronales

Los avances tradicionales en el estado del arte en la detección de acciones humanas utilizan complejos procedimientos de extracción de características diseñados para resolver el problema mayormente en ambientes controlados. Sin embargo, las redes neuronales convolucionales pueden procesar directamente los datos de entrada sin la necesidad de un mayor tratamiento. Es decir, la potencia de las redes neuronales radica en la posibilidad de extraer características de forma automatizada (Shuiwang Ji, Wei Xu, Ming Yang, & Kai Yu, 2013).

4.2.1. Redes convolucionales 3D

Las redes neuronales convolucionales procesan entradas bidimensionales, es decir, pueden procesar imágenes sin la necesidad de incluir la dimensión del tiempo. Para poder adaptar dichas redes al procesamiento de video, Shuiwang propone una arquitectura de red convolucional tridimensional que procese las secuencias de videos en bloques de 7 frames de profundidad por el ancho y alto del video (60 x 40 píxeles). Además, no extrae los frames de forma consecutiva sino dejando un frame de espacio; es decir, para la posición 0, aplica el kernel sobre los frames -6, -4, -2, 0, 2, 4 y 6. Los experimentos son realizados sobre el conjunto de datos TRECVID 2008, el cual agrupa 49 horas de video obtenido por 5 cámaras de seguridad del London Gatwick Airport. Las clases son definidas como tres acciones: CellToEar, ObjectPut y Pointing. Finalmente, la precisión general alcanzada es de 90.2% sobre la base de datos de video mencionada.

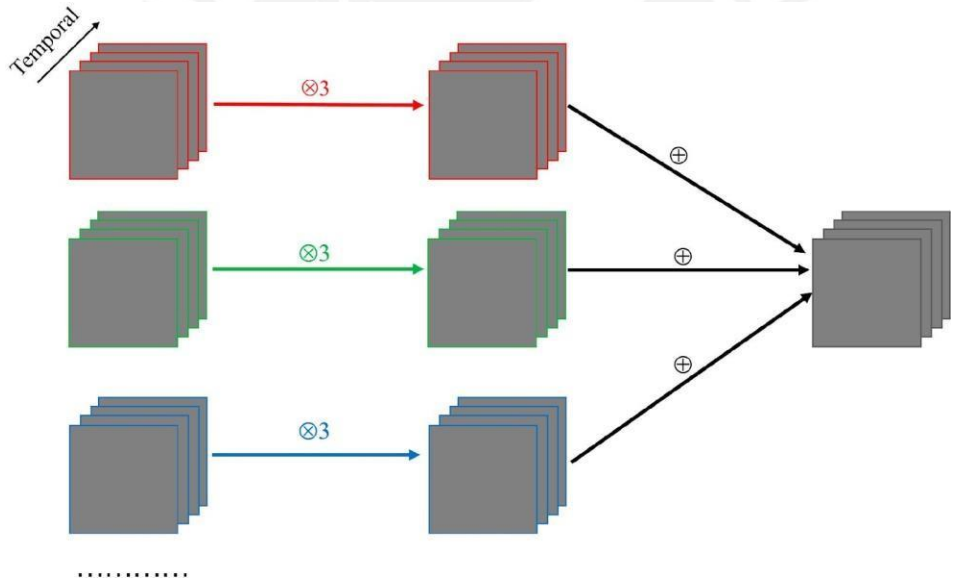

Figura 9: Convolución 3D y mapa de características 3D en imágenes espacio-temporales (Shuiwang Ji et al., 2013).

Por otro lado, otro investigador que utiliza las redes neuronales 3D para identificar acciones humanas es Chunhui Ding, quien aplica un enfoque muy similar al de Shuiwang para poder construir el modelo neuronal (Chunhui Ding, Shouke Fan, Ming Zhu, Weiguo Feng, & Baozhi Jia, 2014). Dicha arquitectura, tal como se aprecia en la Figura 10, procesa una entrada de 60 x 90 píxeles y una profundidad de 40 frames en la capa de entrada, luego posee una capa de convolución que extrae 7 mapas de

características de 2 dimensiones, seguida de una capa de submuestreo, esta secuencia se repite dos veces para luego añadir dos capas totalmente conectadas al final de la red tal como se muestra en la Figura 10. La base de datos utilizada fue el Hockey Dataset de la NHL, las clases son dos: fight y non-fight y el tamaño de la muestra es de 1000 clips de video. Finalmente, la precisión máxima lograda fue de 91% con una tasa de apredizaje de 1.0 y 20 épocas. (Chunhui Ding et al., 2014).

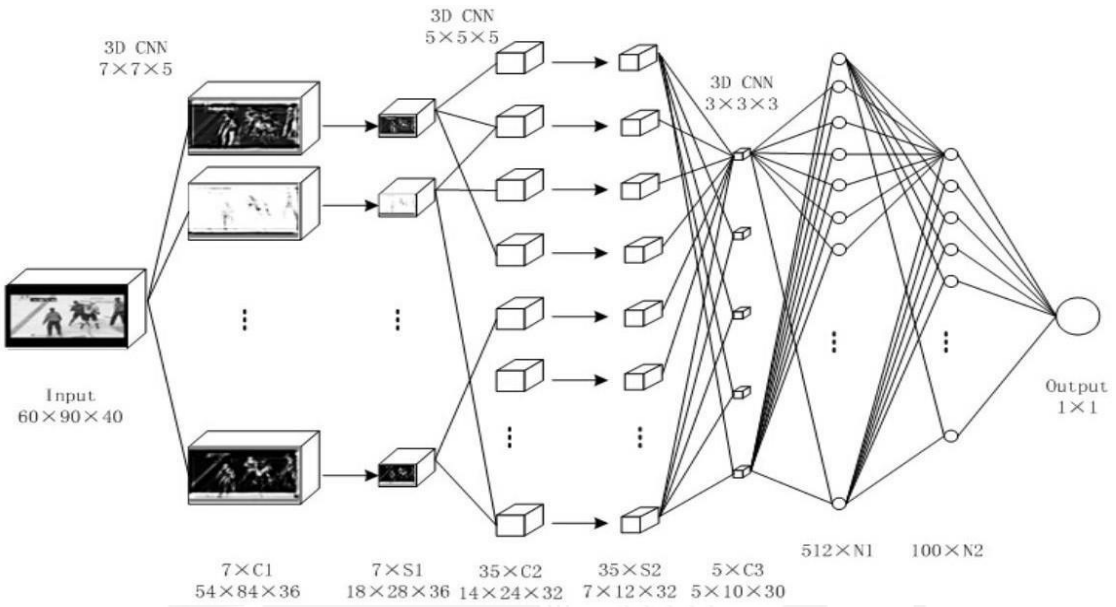

Figura 10: Arquitectura de red convolucional 3D para detección de violencia en video (Chunhui Ding et al., 2014).

4.2.2. Fusión de información de tiempo

En 2014, Andrej Karpathy analizó diferentes técnicas de fusión de información de tiempo para la clasificación de video utilizando una base de datos de grandes dimensiones. Las técnicas exploradas fueron del tipo *Single-frame, Early Fusion, Late Fusion y Slow Fusion,* tal como se muestra en la Figura 11. En la primera, se analiza cada frame del video individualmente, alcanzando un *accuracy* categórico de 59.3% y una tasa de acierto categórico *top*-5 de 77.7%. Luego, en la técnica de fusión temprana, que procesa un bloque tridimensional de video que utiliza el tiempo T ─medido en cantidad de frames─ como tercera dimensión, logrando un *accuracy* categórico de 57.7% y una tasa de acierto categórico[2] top-5 de 76.8% para la predicción. Por otro lado, la fusión tardía procesa dos frames separados entre sí, de

[2] La *tasa de acierto categórico top*-k indica la fracción de muestras de validación que contiene al menos una de las etiquetas de clase correctas en las k predicciones superiores o con mayor valor.

forma independiente ─similar al single frame─, para luego combinar ambos flujos en una capa totalmente conectada, obteniendo un *accuracy* categórico de 59.3% y una tasa de acierto categórico *top*-5 de 78.7%. Finalmente, se propuso una arquitectura de fusión lenta que procesa de forma paralela cuatro bloques de frames tridimensionales de entrada con una profundidad T=4, para clips de 10 frames de longitud, estas cuatro entradas se fusionan en dos cadenas de convolución, normalización y pooling obteniendo un *accuracy* categórico de 60.9% y una tasa de acierto categórico *top*-5 de 80.2% (Karpathy et al., Jun 2014).

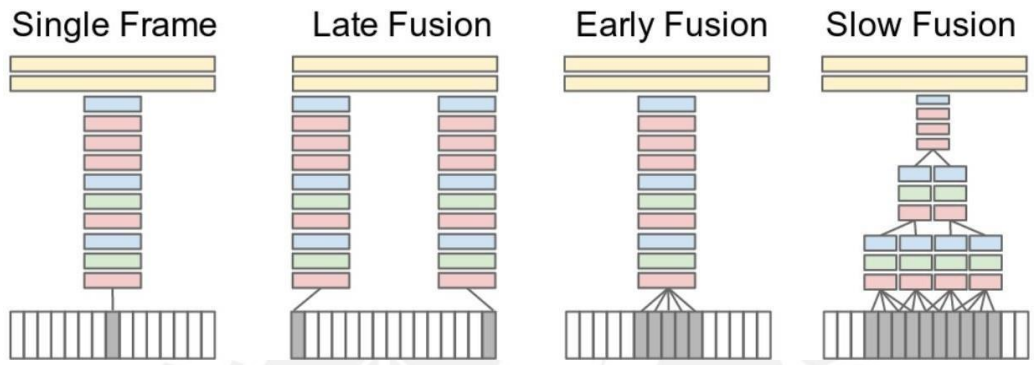

Figura 11: Enfoques explorados para fusionar información temporal donde las tres primeras capas son convolución, normalización y *pooling (Karpathy et al., Jun 2014)*.

4.2.3. Secuencias paralelas de frames y de flujo óptico

Uno de los enfoques utilizados para la clasificación de violencia en video la realizó Qi Dai en el MediaEval 2015, donde exploró diversas combinaciones de métodos de aprendizaje profundo con características extraídas manualmente. Primero, el autor entrena una arquitectura de red *AlexNet (Krizhevsky, Sutskever, & Hinton, 2012)* con un subconjunto de clases de *ImageNet* seleccionadas especialmente para la detección de violencia, para luego utilizar un clasificador de máquina de vectores de soporte (SVM), alcanzando una precisión media (MAP) de 23.5%. Adicionalmente, el investigador utiliza dos redes convolucionales paralelas para extraer características de los frames individuales y de los flujos ópticos de movimiento, aplicando un clasificador SVM sobre los conjuntos de características extraídos, logrando una MAP de 29.5%. Por último, otro de los modelos utilizados incluye una red de larga memoria a corto plazo (LSTM), tal como se aprecia en la Figura 12, pre-entrenada con el conjunto de videos UCF-101, que se aplica sobre las características extraídas de los dos flujos ─frames individuales y flujo óptico de movimiento─ alcanzando una MAP de 29.6% (Dai et al., 2015).

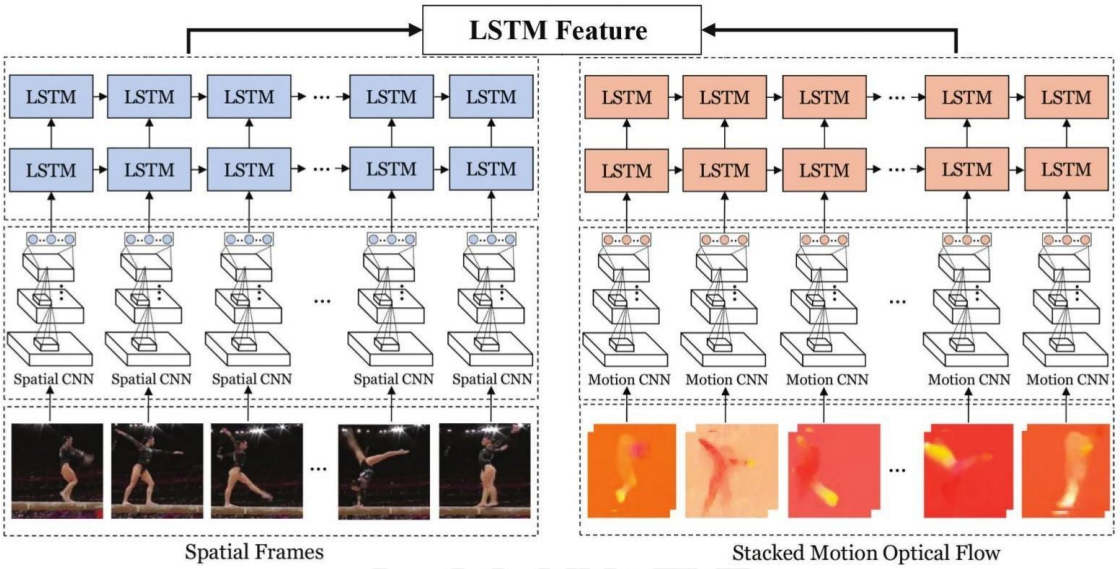

Figura 12: Modelo de red de dos secuencias que utiliza una LSTM para extraer características entre frames distantes (Dai et al., 2015).

Por otro lado, Simonyan y Zisserman, utilizan una arquitectura dos secuencias de redes convolucionales casi idénticas, donde una predice individualmente cada frame y la otra realiza la predicción a partir de una una pila de 10 frames de flujo óptico calculados externamente, ambas salidas son promediadas (Simonyan & Zisserman, 2014). La red que procesa el flujo óptico de frames corriente posee una capa convolucional de inicial adaptada con dos veces más canales de entrada que cuadros de flujo ─esto debido a que el flujo tiene dos canales, horizontal y vertical. Finalmente, la red espacial fue pre-entrenada usando el la base de datos *ImageNet,* para dar como resultado, aplicado en el *dataset* UCF-101, una precisión de 86.2% usando el promedio como método de fusión de ambas redes y un 87% usando SVM (Simonyan & Zisserman, 2014).

4.2.4. Redes convolucionales y LSTM

El alto rendimiento de las redes de clasificación de imágenes permite que sea atractivo intentar reutilizarlas con el mínimo cambio posible para el video. Esto se puede lograr extrayendo características de forma independiente de cada fotograma para luego agrupar cada predicción en una secuencia. Esto se realiza de forma similar a la bolsa de palabras que se basa en el modelado de imágenes; sin embargo, para no ignorar la dimensión temporal se realiza un procesamiento adicional. Por ello, para incluir esta dimensión adicional, se añade una capa recurrente al modelo, como, por ejemplo, una

LSTM, que puede codificar el estado, y capturar el orden temporal y las dependencias de largo plazo. En el modelo que analiza Carreira y que se muestra en la Figura 13, se aplica una normalización de lotes después de la última capa de average polling de Inception-V1, con 512 unidades ocultas, luego se agrega la capa LSTM. Finalmente se añade una capa completamente conectada en la parte superior para el clasificador (Carreira & Zisserman, 2017).

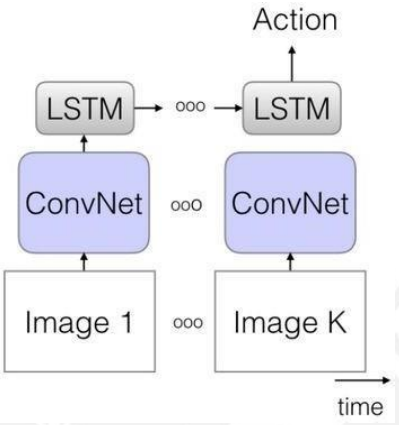

Figura 13: Arquitectura de predicción de dos pasos: extracción de características de frames individuales y predicción de secuencia extraída (Carreira & Zisserman, 2017).

5. Diseño de la Solución

Para la clasificación de acciones se utilizará una red convolucional para la extracción de características de cada frame y una capa LSTM para la captura de las características temporales, tal como se aprecia en la Figura 13. Esta arquitectura será aplicada a dos bases de datos: UCF-101 y Virat 2.0. Los resultados serán comparados al final de esta sección.

5.1. Bases de datos

Debido a la dificultad de poder encontrar una base de datos de acciones violentas en capturas de videovigilancia se propone utilizar un conjunto público de videos de acciones humanas, UCF-101 y otro conjunto, también público, de acciones humanas captadas en secuencias de videovigilancia, Virat 2.0.

5.1.1. UCF-101

Es una base de datos con uno de los más grandes *datasets* de acciones humanas disponible. Consta de 101 clases, 13000 clips y 27 horas de de video. Además, los videos tienen una duración media de 7.21 segundos, siendo el más extenso de 71.04 s y el más corto de 1.06 s. Donde cada clip tiene una tasa de muestreo de 25 cuadros por segundo y una resolución de 320x240 píxeles. Además, las muestras tienen como característica principal el haber sido capturadas en escenas realistas y con fondos en movimiento y estáticos. Complementariamente a ello, se observa en este *dataset* que todos los videos tienen un enfoque o plano que encuadra al humano y a la acción realizada. Finalmente, tal como se aprecia en la Figura 14, las acciones se clasifican en los siguientes tipos: interacción humano-objeto (cuadros con borde de color azul), solo movimiento corporal (borde de color rojo), interacción humano-humano (borde de color morado), tocando instrumentos musicales (borde de color turquesa) y deportes (borde de color verde) (Soomro, Zamir, & Shah, 2012).

Figura 14: Las 101 acciones incluídas en el *dataset* UCF-101 diferencias por tipo de acción según el color del marco (Soomro et al., 2012).

5.1.2. Virat 2.0

Es un volumen de datos de video de vigilancia de grandes dimensiones diseñado para evaluar el rendimiento de los algoritmos de reconocimiento de eventos en escenas realistas. El conjunto de datos incluye videos recopilados tanto de cámaras terrestres fijas como de vehículos aéreos en movimiento. Además, consta de 16 escenas con acciones que ocurren naturalmente, sin actores, en videos capturados continuamente del mundo real. Para cumplir con los objetivos de esta investigación se propone utilizar el subconjunto de videos terrestres que incluye tomas exclusivas de cámaras de videovigilancia. A diferencia del *dataset* UCF101, estos videos solo contienen tomas

con planos generales o abiertos, los cuales capturan el íntegro de la escena sin realizar encuadres en los humanos ni en sus acciones. El conjunto de datos completo consta de 23 tipos de eventos, de 10 a 1500 muestras por tipo, resolución de 1920x1080 píxeles, 17 diferentes escenas, y 29 horas de video. Las acciones contenidas por tipo son las siguientes: (Oh et al., 2011)

- Para solo una persona: caminar, correr, pararse, tirar, gesticular, cargar, merodear, recoger.
- Persona y vehículo: entrar o salir del vehículo, abrir o cerrar el maletero, cargar, descargar, dejar caer, andar en bicicleta.
- Persona e instalaciones: entrar o salir de la instalación.

De este conjunto se seleccionarán los doce tipos de acción contenidos en el subconjunto de videos terrestres de VIRAT 2.0:

1. Persona cargando un objeto a un vehículo
2. Persona descargando un objeto de un automóvil / vehículo
3. Persona abriendo un vehículo / maletero delcoche
4. Persona cerrando un vehículo / maletero delcoche
5. Persona entrando a un Vehículo
6. Persona saliendo de un Vehículo
7. Persona gesticulando
8. Persona excavando
9. Persona llevando un objeto
10. Persona corriendo
11. Persona ingresando a una instalación
12. Persona saliendo de una instalación

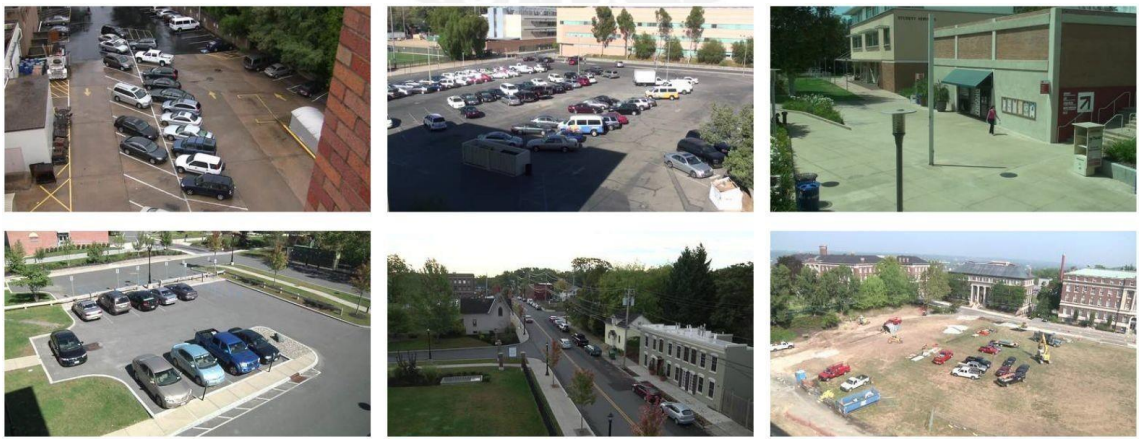

Figura 15: Seis escenas contenidas en la base de datos VIRAT

Este *dataset*, a diferencia del anterior, provee un archivo con metadatos que incluye las anotaciones de los eventos contenidos en cada archivo de video. Esto debido a que las 1555 acciones de este subconjunto de datos están distribuídas en 329 archivos de video. La estructura del archivo con metadatos de eventos, para cada una de las 10 columnas, es el siguiente:

1. Identificador del evento.
2. Tipo de evento.
3. Duración del evento en cantidad de frames.
4. Frame de inicio del evento: base cero.
5. Frame final del evento.
6. Coordenada X del box que delimita la acción dentro del frame.
7. Coordenada Y del box que delimita la acción dentro del frame.
8. Ancho del box que delimita la acción dentro del frame.
9. Altura del box que delimita la acción dentro del frame.
10. Número total de objetos involucrados.

Finalmente, para poder realizar la extracción de los eventos de este *dataset* fue necesaria la construcción de algunos scripts para el preprocesamiento de los videos.

5.2. Métricas

Para poder evaluar los resultados se utilizarán las siguientes métricas:

5.2.1. Definiciones

Se tienen las siguientes definiciones para los resultados obtenidos de la clasificación de acciones, considerando que, cuando se evalúan estos valores para una clase A esta se toma como verdadera y todas las demás (B, C, D, etc.) se consideran falsas:

- Verdadero positivo (TP) = la cantidad de secuencias de video cuya acción A identificada SÍ es la acción correcta.
- Falso positivo (FP) = la cantidad de secuencias de video cuya acción A identificada NO es la acción correcta. Error de tipo I
- Verdadero negativo (TN) = la cantidad de secuencias de video cuya acción identificada (B, C, D, etc.) SÍ es la acción correcta.
- Falso negativo (FN) = la cantidad de secuencias de video cuya acción identificada (B, C, D, etc.) NO es la acción correcta. Error de tipo II

Tabla 1: Matriz de confusión y distribución de las definiciones anteriores dentro de las celdas de la matriz..

Clase \ Clase predicha \ verdadera	**A**	**B, C, D, etc.**
A	Verdadero positivo (TP)	Falso positivo (FP)
B, C, D, etc.	Falso negativo (FN)	Verdadero negativo (TN)

5.2.2. Accuracy y Categorical accuracy

Ambas métricas, debido a la naturaleza de los datos multiclase utilizados en esta investigación, significan lo mismo. Esta mide la exactitud del algoritmo a partir de la cercanía entre el valor de la cantidad medida y el valor de cantidad real de un mensurando (Joint Committee for Guides in Metrology, 2008). En este caso, la medida es equivalente a la media de las predicciones de la clase de cada dato en contraposición con la clase real o *ground truth* de cada dato.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad (2)$$

5.2.3. Precisión

La precisión es la relación entre número de verdaderos positivos y la suma de los verdaderos positivos y los falsos positivos. Es decir, es la capacidad del clasificador de no etiquetar como positiva una muestra que es negativa.

$$precisión = \frac{TP}{TP + FP} \qquad (3)$$

5.2.4. Recall

La precisión es la relación entre número de verdaderos positivos y la suma de los verdaderos positivos y los falsos negativos. Es decir, es la capacidad del clasificador para encontrar todas las muestras positivas.

$$recall = \frac{TP}{TP + FN} \qquad (4)$$

5.2.5. F1 Score

La puntuación F-beta o F1 score, es la media armónica ponderada de la precisión y el recall. El mejor puntaje para F-beta es 1 y el peor es 0.

$$F - beta = \frac{2TP}{2TP + FP + FN} \quad (5)$$

5.3. Experimentos

La siguiente sección muestra los experimentos realizados utilizando la arquitectura seleccionada sobre los dos conjuntos de datos.

5.3.1. Enfoque de red convolucional Inception-v3 y LSTM sobre UCF-101

Este enfoque utiliza la arquitectura de dos redes neuronales convolucionales para realizar la tarea de clasificación. La primera red es usada como un extractor de características y la segunda red realiza la clasificación del conjunto de características a lo largo de un tiempo determinado.

Dentro de este contexto se selecciona el modelo Inception-v3 (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2016) para poder extraer las características de cada frame del video en la primera parte del procesamiento. Este modelo es el resultado de una serie de replanteamientos de la estructura inicial del modelo de visión computacional GoogLeNet, después de Inception-v1 (Szegedy et al., 2015) e Inception-v2 (Ioffe & Szegedy, 2015). Además, el modelo Inception-v3 fue entrenado con el conjunto de datos de ImageNet, el cual contiene un total de 1000 clases de imágenes naturales, una tasa de error top-5 es 3.5%, y una tasa de error top-1 de 17,3%. La arquitectura de esta red está conformada por una serie de convoluciones y submuestreo iniciales (tal como se aprecia en la parte izquierda de la Figura 16) para luego dar paso a diferentes módulos inception que, mediante diferentes convoluciones, submuestreos y concatenaciones, buscan obtener la mayor cantidad de características de los datos de entrada (los cuales se aprecian en la parte central de la arquitectura mostrada en la Figura 16), finalmente la red culmina con una capa completamente conectada y una capa donde se aplica el clasificador.

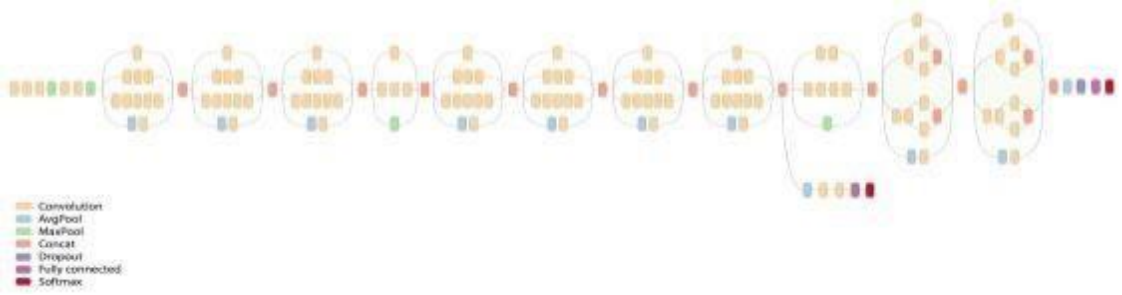

Figura 16: Arquitectura de red del modelo Inception v3(Szegedy et al., 2016)

5.3.1.1. Preprocesamiento de datos

Antes de iniciar el proceso, una vez obtenido el conjunto de datos, se extraen los archivos en tantas carpetas como acciones se tiene, es decir, en ciento un directorios dentro de los cuales se encontrarán los videos correspondientes a las acciones mencionadas.

Figura 17: Estructura de carpetas de los tipos de acción del *dataset* UCF101

Luego de ello, se debe extraer los frames de cada uno de los videos y guardarlos como archivos dentro de subcarpetas con el nombre del video en cuestión. Para ello se hace uso de la librería *ffmpeg (Bellard, 2005)* y se repite el proceso de extracción de frames sin perder la estructura de carpetas. Además, se debe considerar que los videos están codificados a una tasa de 25 frames por segundo.

A continuación, se realiza una distribución aleatoria de las muestras entre el conjunto de entrenamiento y el conjunto de validación. En dicha distribución, que se realizó una vez, se utilizó un ratio de 70% de muestras para entrenamiento y 30% para validación. En este punto los datos se encuentran balanceados: con una media de muestras totales por acción de 131.88 y una desviación estándar de 18.31 con un mínimo de 100 y un máximo de 167 muestras. En la Tabla 2 se aprecia la distribución de muestras de algunas de las acciones del *dataset.*

Tabla 2: Distribución de la cantidad de muestras por acción para entrenamiento y validación en un ratio de 70% y 30% alcanzando un mínimo de 100 y un máximo de 167 videos por acción humana. En la tabla se muestra solo 22 de las 101 acciones.

Action	train	test	total
ApplyEyeMakeup	101	44	145
ApplyLipstick	82	32	114
Archery	104	41	145
BabyCrawling	97	35	132
BalanceBeam	77	31	108
BandMarching	112	43	155
BaseballPitch	107	43	150
Basketball	99	35	134
BasketballDunk	94	37	131
BenchPress	112	48	160
Biking	96	38	134
Billiards	110	40	150
BlowDryHair	93	38	131
BlowingCandles	76	33	109
. . .	. . .	. . .	. . .
TrampolineJumping	87	32	119
Typing	93	43	136
UnevenBars	76	28	104
VolleyballSpiking	81	35	116
WalkingWithDog	87	36	123
WallPushups	95	35	130
WritingOnBoard	107	45	152
YoYo	92	36	128
Total	**9537**	**3783**	**13320**

De forma seguida, se procede con un filtrado de muestras por longitud de video. Esto debido a que se comparan secuencias de frames y no frames individuales, por tanto, el espacio temporal, medido en frames, entre cada una de las muestras debiera ser comparable. Para lograr esto, más adelante se observará que se estandariza la cantidad de frames por muestra al reducirla a 40. Sin embargo, al realizar esta reducción se termina perdiendo información, por ello, si las secuencias de video son muy largas, esta pérdida es tan dramática que se afectaría la capacidad de realizar la clasificación de la acción. Por tanto, se decide filtrar los videos con una longitud menor a 40 y además los que tienen una longitud mayor a 300. Este proceso ocasiona un desbalance que afecta de diferente forma a la lista de acciones a procesar, es decir, algunas acciones pierden mayor cantidad de muestras mientras que otras no pierden ningún número significativo de muestras. Como se puede apreciar en la Tabla 3, las acciones con mayor porcentaje de pérdida de muestras son: *JumpRope, RockClimbingIndoor, PlayingSitar, PommelHorse y Rowing,* además un total de 36 acciones tuvieron una pérdida inferior al 10% de muestras y 40 acciones no tuvieron pérdida alguna. En consecuencia, la distribución de muestras disponibles por acción humana concluye con las siguientes estadísticas: una media de muestras totales por acción de 118.95 y una desviación estándar de 26.87 con un mínimo de 15 y un máximo de 167 muestras.

Tabla 3: Lista de 12 acciones con mayor porcentaje de pérdida de muestras, debido al filtrado por longitud mayor a 40 y menor a 300 frames, ordenadas de mayor a menor pérdida.

Action	train	test	total	samples lost %
JumpRope	10	5	15	90%
RockClimbingIndoor	25	6	31	78%
PlayingSitar	28	18	46	71%
PommelHorse	30	12	42	66%
Rowing	55	11	66	52%
PlayingDaf	71	16	87	42%
SoccerJuggling	68	22	90	39%
Billiards	76	20	96	36%
PlayingFlute	67	37	104	33%
PlayingDhol	85	26	111	32%
SalsaSpin	68	30	98	26%
PlayingTabla	60	24	84	24%

En este primer experimento no se ejecuta ninguna estrategia de balanceo de datos con la finalidad de evaluar la potencia del método de clasificación en sí misma para el conjunto resultante. Como se aprecia en el gráfico de la Figura 18, las acciones con porcentaje de pérdida de muestras mayor a 25% son solamente 11 de las 101. Estas 11 acciones so pueden apreciar con mayor detalle en la Tabla 3.

Figura 18: Porcentaje de muestras no filtradas por cantidad de frames mayor a 40 y menor a 300.

Finalmente, se genera un archivo CSV con la siguiente estructura: tipo de muestra (entrenamiento / validación), nombre de acción, nombre de video y cantidad de frames. Este archivo servirá de entrada para la fase de procesamiento de los datos que se detalla en la siguiente sección.

5.3.1.2. Procesamiento de datos

Este proceso inicia con la carga de los frames, previamente extraídos, en una lista. Para el *dataset* UCF101 estas listas tienen longitudes que van desde 29 hasta 1776 frames antes del filtrado y de 40 a 300 frames después del filtrado. Tal como se aprecia en la Figura 19, cada frame se almacena como un archivo de imagen en la unidad del computador.

Figura 19: Captura de pantalla con una pequeña muestra de los frames extraídos de un video del *dataset* UCF101 del tipo de accón BlowingCandles.

Como siguiente paso, se tiene la reducción de la dimensión de la lista de frames con la finalidad de estandarizar las listas a un número arbitrario n que, en este caso, será de 40 frames. Esta reducción se consigue al extraer los j-ésimos frames de la lista de entrada, donde la posición j se calcula como sigue:

$$int(\frac{len(inputList)}{n}) * i,\ i : 0..n-1 \qquad (6)$$

Considerando que la lista de entrada tendrá una dimensión entre 40 y 300 frames, la lista de salida se construirá tomando el j-ésimo frame, donde j es, a lo más, múltiplo de siete. En la siguiente figura se observa un segmento de las listas de entrada y de la lista resultante, considerando el tamaño de la entrada de 120 frames donde j es múltiplo de 3. En la Figura 20 se aprecia que, la reducción del tamaño de la lista no genera una pérdida de información importante.

Figura 20: Segmento de la lista de entrada de 120 frames y de la lista de salida de 40 frames.

Luego de la reducción de la longitud del video se debe proceder con la extracción de características de cada frame. Para esto se recuerda que la arquitectura que se utiliza posee dos grandes secciones con dos redes neuronales para diferentes fines. En la primera se procesa cada frame de la lista de salida de forma individual, con la finalidad de obtener el vector de características del frame, tal como lo sugiere Carreira (Carreira & Zisserman, 2017). Es decir, se extraen las características individuales de cada frame utilizando un modelo de red convolucional, en este caso, Inception v3. Mientras que la segunda red es utilizada para la clasificación. En otras palabras, ambas redes compartirán información a partir de la salida de una que servirá como entrada para la siguiente red. La pregunta que surge en este punto es acerca de qué capa de la arquitectura de la primera red seleccionar para poder extraer las características. Para tomar esta decisión, se utiliza la recomendación de Lan (Lan, Zhu, & Hauptmann, 2017) de extraer las características de la última capa pooling (tal como se aprecia en la Figura 21 donde la red Inception v3 está truncada en la última capa *Average Pooling* de color celeste), la cual alcanza el mayor *accuracy* en el *dataset* UCF101. Aunque, considerando que el autor utiliza otra arquitectura para la clasificación de la línea temporal. Cabe mencionar que para llegar a esta conclusión, Lan compara los resultados de extraer las características de la última capa *fully connected, global polling, 5b, 5a y 4e,* aplicada sobre su arquitectura y encuentra que al extraer las características de la capa global pooling se obtiene el mejor accuracy. Como se puede apreciar en la Tabla 4, que muestra un segmento exacto de la arquitectura utilizada, la última capa de la arquitectura Inception-v3 usada es la capa *GlobalAveragePooling2*.

Tabla 4: Detalle de las últimas dos capas de la arquitectura de la red Inception V3 usada para la extracción de características

```
...
__________________________________________________________________________________________________
mixed10 (Concatenate)           (None, None, None, 2 0           activation_86[0][0]

                                                                 mixed9_1[0][0]

                                                                 concatenate_2[0][0]

                                                                 activation_94[0][0]

__________________________________________________________________________________________________
avg_pool (GlobalAveragePooling2 (None, 2048)     0       mixed10[0][0]
==================================================================================================
Total params: 21,802,784
Trainable params: 21,768,352
Non-trainable params: 34,432
```

Tabla 5: Segmento de un vector de características extraído de un solo frame.

```
[3.7887284159660339 36e-01 1.9322371482849121 09e-01 2.0402541756629943 85e-01 3.9272427558898925 78e-01 6.7343994975090026 86e-02 3.8289174437522888 18e-01
4.5383530855178833 01e-01 1.4000605046749114 99e-01 1.3199342787265777 59e-01 6.0884678363800048 83e-01 6.0642683506011962 89e-01 1.2551119923591613 77e-01
1.9488963484764099 12e-01 7.1752411127090454 10e-01 3.0636763572692871 09e-01 5.0938260555267333 98e-01 ...]
```

Este proceso de extracción de características culmina con la concatenación de los 40 vectores extraídos y la serialización en un solo archivo que luego será consultado en la siguiente sección o fase. En la Tabla 5 se aprecia un segmento del vector de características. Además, el tiempo de procesamiento utilizado en esta parte del proceso es de 51:50 minutos para los 12023 videos del *dataset* UCF101 que se obtienen luego del filtro. En la Figura 21 se aprecia un diagrama de la primera sección del proceso para un solo video, donde se aprecia, para un video, los pasos de 1) extracción de frames, 2) reducción de dimensionalidad de frames a un total de 40, 3) extracción de frames, 4) obtención de vector de características del frames utilizando la red Inception v3 y 5) la concatenación de los frames extraídos de un video en un solo archivo de texto hasta terminar de procesar los 40 frames del video.

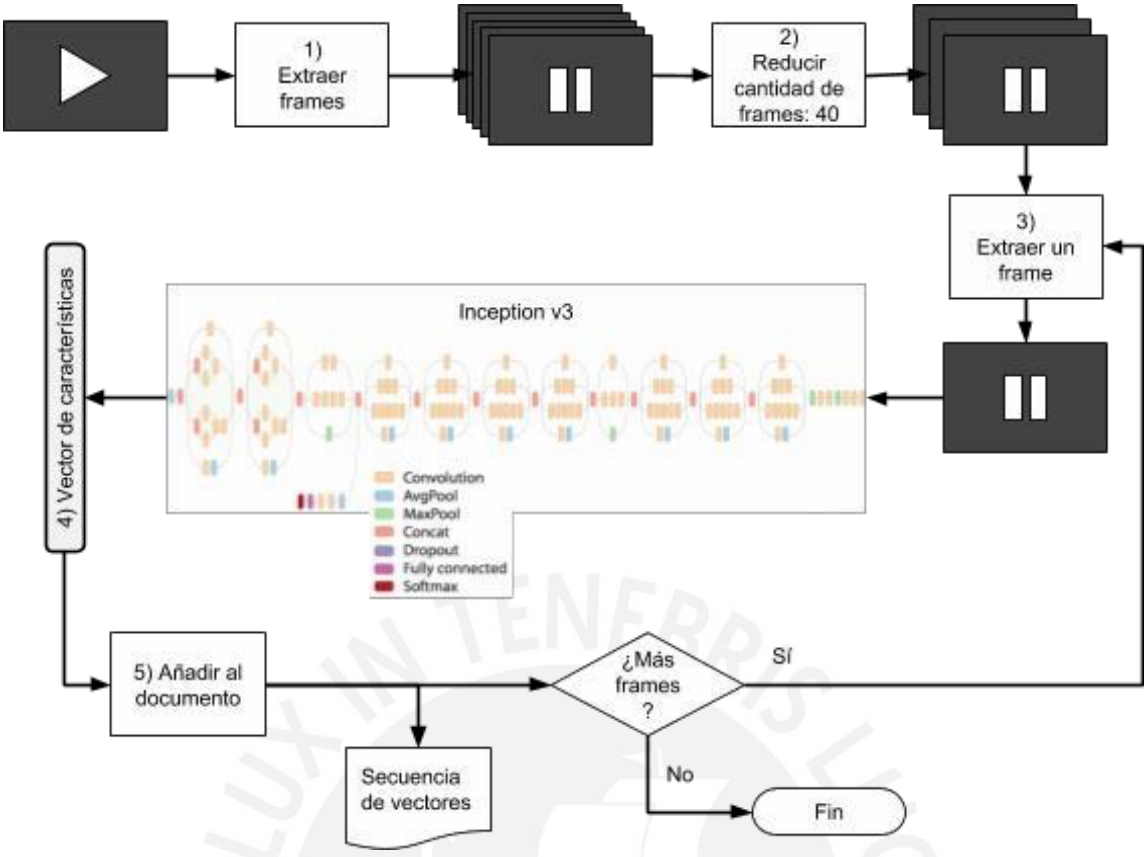

Figura 21: Primera sección del proceso: Extracción de frames, reducción a 40 frames y extracción de características por frame para un solo video.

La siguiente sección del procesamiento se enfoca en utilizar los vectores de características extraídos de los frames de cada video con la finalidad de entrenar una red con capacidad de memoria. Para esto se construye una arquitectura basada en una red LSTM la cual procesa cada uno de los videos conformados por vectores de longitud 40 (uno por cada frame de video) y con vectores de características de longitud 2048 (que es el vector obtenido de la capa *Global Average Polling* de la red Inception-V3). La salida obtenida de la capa LSTM se procesa usando una capa *Flatten* de modo que se obtiene un vector de una dimensión y de longitud 81920. Finalmente, para la clasificación, se introduce este vector en una capa completamente conectada, *Dense*, de 512 unidades y luego en otra capa *Dense* con una función de activación Softmax para la clasificación. La arquitectura de esta red se puede apreciar en la Figura 22. Donde la capa LSTM posee una salida de la forma (None, 40, 2048) y un total de 33,562,624 parámetros, la capa Flatten posee una salida de la forma (None, 81920), la primera capa Dense posee una salida de la forma (None, 512) y un total de 41,943,552 con Dropout, y la última capa Dense posee una salida de la forma (None, 101) con un total de 51,813 parámetros. Esta red tiene un total de 75,557,989

parámetros entrenables.

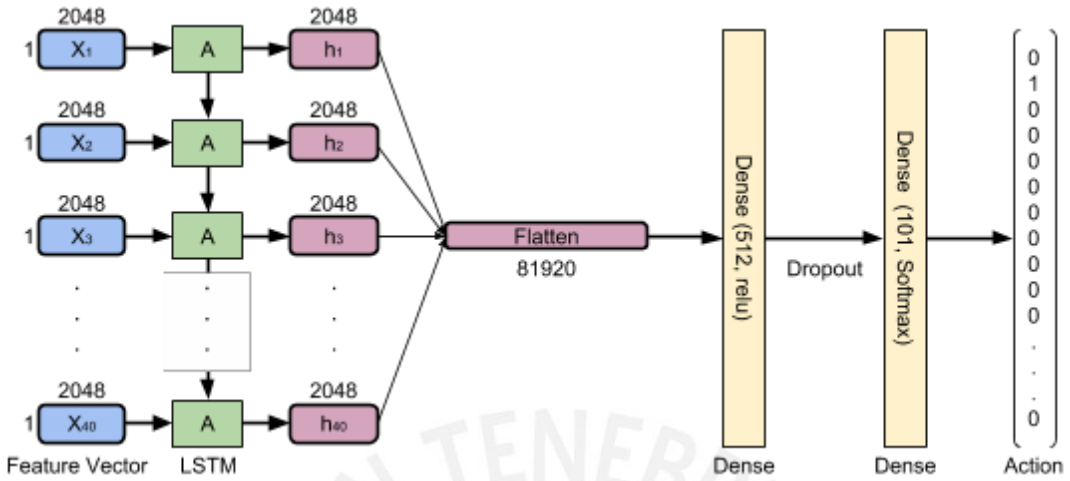

Figura 22: Arquitectura de la red con memoria para la clasificación de las 101 acciones del *dataset* UCF101.

5.3.1.3. Resultados

Luego de introducir el dataset UCF101 a la arquitectura antes descrita, se obtuvo los siguientes resultados: El entrenamiento de la red LSTM, utilizando el dataset UCF101, y la validación tardaron en conjunto un total de 31241.3 segundos, es decir, ocho horas y cuarenta minutos.

Además, se configuró el algoritmo para que realizara 1000 épocas con un *batch size* de 128, adicionalmente se utilizó un optimizador Adam (Kingma & Ba, 2014) con un learning rate de 0,002478752 (es decir de e^{-6}) y una función de pérdida de tipo entropía cruzada categórica. Con los parámetros mencionados anteriormente se obtienen los siguientes resultados al finalizar el entrenamiento y validación:

- Para los datos de entrenamiento:
    - loss function: 0.0673
    - categorical accuracy: 0.9812
    - top 5 categorical accuracy: 0.9993
- Para los datos de validación:
    - loss function: 1.2410
    - categorical accuracy: 0.7510
    - top 5 categorical accuracy: 0.9289

De estos valores, en primer lugar, se aprecia que la función de pérdida en los datos de entrenamiento alcanza un valor muy bajo, es decir, la función indica que el modelo es capaz de aprender. Esto se corrobora en la Figura 23, donde se observa la gráfica de la función de pérdida en los datos de entrenamiento de color azul y en los datos de validación de color anaranjado. Además, se puede observar que la función en los datos de validación comienza a perder capacidad de generalización a partir de la época 300 y por tanto la función de pérdida comienza a incrementar suvalor.

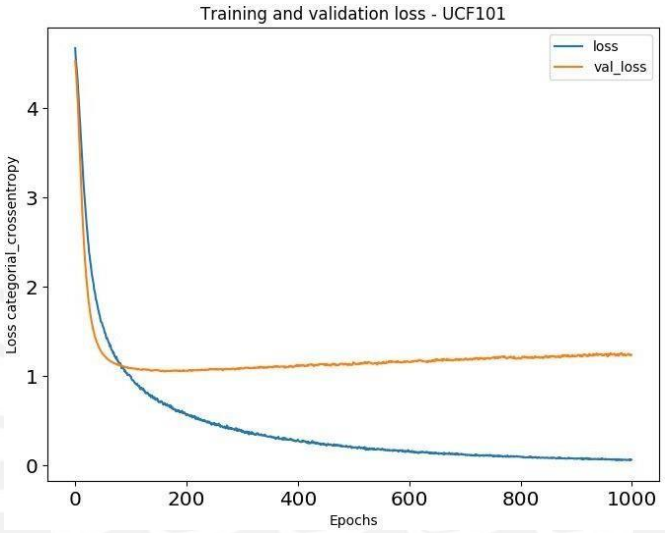

Figura 23: Gráfica de la función de pérdida para los datos de entrenamiento (de color azul) y para los datos de validación (de color anaranjado) del modelo de red con memoria para el conjunto de datos UCF101.

En segundo lugar, se puede apreciar que el accuracy categórico en los datos de entrenamiento alcanza la cifra de 98.12% mientras que en lo datos de validación alcanza el valor de 75.12%. De estos valores, junto con el valor final de la función de pérdida para los datos de entrenamiento, se deduce que el modelo comienza a sobreentrenarse o hacer *overfitting.* Además de lo anterior, la tasa de acierto categórico top-5 alcanza el valor de 99.93% para los datos de entrenamiento y de 92.89% para los datos de validación lo cual se aprecia en la Figura 24. Estos valores son lo suficientemente aceptables para el objetivo general que se persigue: ser de apoyo a la labor de videovigilancia.

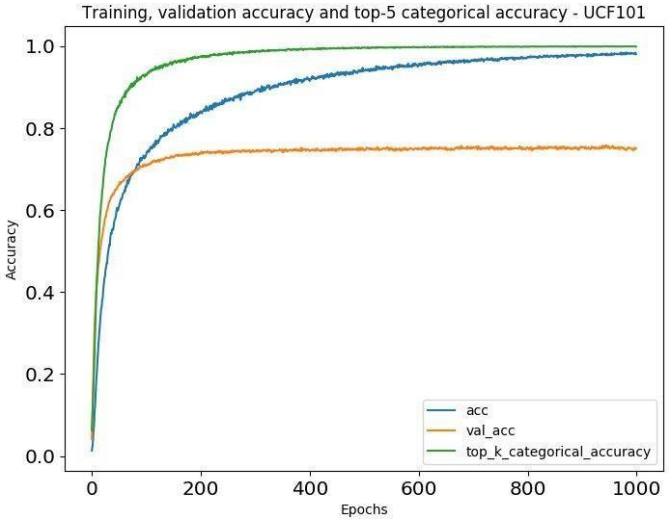

Figura 24: Gráfica de la precisión categórica para los datos de entrenamiento (de color azul) y para los datos de validación (de color anaranjado) del modelo de red con memoria para el conjunto de datos UCF101. Además se aprecia la tasa de acierto categórico top-5.

Además, se muestran otras métricas que permiten analizar la efectividad del modelo al nivel de las clases individuales. Dichas métricas son la precisión (*precision*), la sensibilidad (*recall*) y la utilidad teórica (*F1 Score*). Si bien es cierto, estas métricas están pensadas para un contexto binario, sí son válidas para un clasificador multiclase si se considera como valor positivo a la clase en evaluación y como negativo a todo el resto de clases en conjunto.

En consecuencia, el modelo tiene una precisión promedio de 76%, siendo las clases *Billiards, PlayingSitar, Typing, PlayingDhol, Bowling, Knitting, PlayingCello* y *TaiChi* las mejor clasificadas con una precisión del 100%. De estas, las dos primeras: *Billiards* y *PlayingSitar* tuvieron una sensibilidad del 100% con un soporte de 20 y 18 muestras de validación respectivamente. Por otro lado, las clases *CricketBowling, SoccerJuggling, JavelinThrow, BodyWeightSquats, JumpRope* y *WallPushups* tuvieron las menor precisión, con un valor mayor a cero en los cuatro primeros casos e igual a cero en los dos últimos tal como se aprecia en las siguientes tablas.

Tabla 6: Lista de las primeras 68 clases con mejor *precision* del conjunto de datos UCF101, con el detalle de la sensibilidad (*recall*) y la utilidad teórica (F1-Score) además del soporte para cada clase en el conjunto de datos de validación.

Action	precision	recall	f1-score	support
Billiards	1	1	1	20
PlayingSitar	1	1	1	18
Typing	1	0.93	0.96	43
PlayingDhol	1	0.92	0.96	26
Bowling	1	0.91	0.95	43
Knitting	1	0.82	0.9	34
PlayingCello	1	0.71	0.83	41
TaiChi	1	0.61	0.76	28
PlayingGuitar	0.98	1	0.99	43
Biking	0.97	1	0.98	32
CuttingInKitchen	0.97	1	0.98	32
SkyDiving	0.97	1	0.98	30
StillRings	0.97	0.91	0.94	32
CliffDiving	0.97	0.85	0.9	39
Fencing	0.97	0.85	0.91	34
PlayingFlute	0.97	0.78	0.87	37
Mixing	0.97	0.76	0.85	45
Diving	0.96	1	0.98	45
IceDancing	0.96	1	0.98	46
PushUps	0.96	0.73	0.83	30
TableTennisShot	0.95	1	0.97	39
Hammering	0.95	0.61	0.74	33
HorseRiding	0.94	0.96	0.95	49
Punch	0.94	0.87	0.91	39
SoccerPenalty	0.94	0.83	0.88	41
BasketballDunk	0.92	1	0.96	35
BreastStroke	0.92	0.96	0.94	25
PlayingPiano	0.92	0.79	0.85	28
PlayingTabla	0.91	0.83	0.87	24
FrontCrawl	0.9	0.81	0.85	32
PommelHorse	0.9	0.75	0.82	12
UnevenBars	0.89	0.86	0.87	28
Kayaking	0.88	0.58	0.7	36
BabyCrawling	0.87	1	0.93	34

Action	precision	recall	f1-score	support
RockClimbingIndoor	0.86	1	0.92	6
GolfSwing	0.85	0.64	0.73	36
BenchPress	0.85	0.48	0.61	48
BlowingCandles	0.84	0.97	0.9	32
WritingOnBoard	0.84	0.84	0.84	45
Skijet	0.83	1	0.91	20
HorseRace	0.83	0.88	0.85	33
Drumming	0.83	0.87	0.85	45
HandstandPushups	0.83	0.71	0.77	28
ApplyEyeMakeup	0.82	0.75	0.79	44
BalanceBeam	0.82	0.45	0.58	31
JumpingJack	0.81	0.59	0.69	37
SumoWrestling	0.8	0.97	0.88	33
Surfing	0.8	0.97	0.88	33
WalkingWithDog	0.8	0.89	0.84	36
Swing	0.8	0.8	0.8	40
BaseballPitch	0.78	0.84	0.81	43
Archery	0.78	0.66	0.71	38
HighJump	0.78	0.49	0.6	37
TrampolineJumping	0.76	0.93	0.84	28
TennisSwing	0.76	0.86	0.81	49
MilitaryParade	0.76	0.79	0.78	33
FrisbeeCatch	0.75	0.81	0.78	37
FieldHockeyPenalty	0.75	0.77	0.76	39
PlayingViolin	0.74	0.88	0.81	26
Skiing	0.74	0.74	0.74	39
PlayingDaf	0.73	1	0.84	16
PizzaTossing	0.73	0.69	0.71	32
ParallelBars	0.72	0.97	0.83	37
Rafting	0.72	0.82	0.77	28
LongJump	0.72	0.46	0.56	39
BandMarching	0.71	0.84	0.77	38
CleanAndJerk	0.71	0.75	0.73	32
BlowDryHair	0.69	0.78	0.73	37

Tabla 7: Lista de las últimas 33 clases con la menor *precision* del conjunto de datos UCF101, con el detalle de la sensibilidad (*recall*) y la utilidad teórica (F1-Score) además del soporte para cada clase en el conjunto de datos de validación.

Action	precision	recall	f1-score	support
HeadMassage	0.69	0.75	0.72	32
ShavingBeard	0.69	0.75	0.72	32
Shotput	0.68	0.54	0.6	46
PoleVault	0.67	1	0.8	38
CricketShot	0.67	0.29	0.4	49
MoppingFloor	0.66	0.74	0.7	31
FloorGymnastics	0.64	0.89	0.74	36
BoxingPunchingBag	0.64	0.81	0.72	48
JugglingBalls	0.64	0.62	0.63	40
RopeClimbing	0.63	0.71	0.67	34
HulaHoop	0.62	0.76	0.68	34
BrushingTeeth	0.62	0.74	0.68	27
HammerThrow	0.6	0.71	0.65	45
YoYo	0.59	0.72	0.65	36
SalsaSpin	0.59	0.67	0.62	30
ApplyLipstick	0.58	0.56	0.57	27
SkateBoarding	0.56	0.74	0.64	31
ThrowDiscus	0.55	0.79	0.65	38
BoxingSpeedBag	0.55	0.73	0.63	30
Basketball	0.53	0.74	0.62	35
VolleyballSpiking	0.49	0.66	0.56	35
Haircut	0.46	0.73	0.56	33
Rowing	0.45	0.45	0.45	11
PullUps	0.44	0.54	0.48	28
Lunges	0.4	0.51	0.45	37
HandstandWalking	0.38	0.09	0.14	34
Nunchucks	0.37	0.21	0.26	34
CricketBowling	0.35	0.33	0.34	36
SoccerJuggling	0.34	0.5	0.41	22
JavelinThrow	0.33	0.52	0.41	31
BodyWeightSquats	0.21	0.17	0.19	30
JumpRope	0	0	0	5
WallPushups	0	0	0	35
avg / total	**0.76**	**0.75**	**0.74**	**3418**

Complementariamente se presenta parte de la matriz de confusión ordenada por el valor de la sensibilidad por clase. En las tablas siguientes se aprecian las 26 clases con mayor sensibilidad (Tabla 8) y las 28 clases con menor sensibilidad (Tabla 9). En la Tabla 8, el valor de sensibilidad más resaltante es el de la clase *RockClimbingIndoor*, el cual posee un 100% de *recall,* sin embargo cuenta con un soporte de 6 unidades. Esto debido a que esta clase perdió un 22% de sus datos totales dado que el resto de muestras excedían los 300 frames.

Tabla 8: Matriz de confusión con las 26 clases con mayor recall del modelo recurrente en la data de validación.

	BabyCrawling	BasketballDunk	Biking	Billiards	CuttingInKitchen	Diving	IceDancing	PlayingDaf	PlayingGuitar	PlayingSitar	PoleVault	RockClimbingIndoor	Skijet	SkyDiving	TableTennisShot	ParallelBars	SumoWrestling	Surfing	BlowingCandles	BreastStroke	HorseRiding	Typing	TrampolineJumping	PlayingDhol	Bowling	StillRings	recall	support
BabyCrawling	34																										1.00	34
BasketballDunk		35																									1.00	35
Biking			32																								1.00	32
Billiards				20																							1.00	20
CuttingInKitchen					32																						1.00	32
Diving						45																					1.00	45
IceDancing							46																				1.00	46
PlayingDaf								16																			1.00	16
PlayingGuitar									43																		1.00	43
PlayingSitar										18																	1.00	18
PoleVault											38																1.00	38
RockClimbingIndoor												6															1.00	6
Skijet													20														1.00	20
SkyDiving														30													1.00	30
TableTennisShot															39												1.00	39
ParallelBars																36											0.97	37
SumoWrestling		1															32										0.97	33
Surfing																		32									0.97	33
BlowingCandles																			31								0.97	32
BreastStroke																				24							0.96	25
HorseRiding																					47						0.96	49
Typing																						40					0.93	43
TrampolineJumping																							26				0.93	28
PlayingDhol																								24			0.92	26
Bowling																									39		0.91	43
StillRings																1										29	0.91	32

Por otro lado, en la Tabla 9, las clases con menor sensibilidad son *WallPushups, JumpRope* y *HandstandWalking*, de estas, las dos primeras poseen un valor igual a cero. Adicionando que *JumRope* posee un soporte de 5 unidades debido a que, junto con *RockClimbingIndoor*, fueron de las clases que perdieron mayor cantidad de muestras por el filtrado. Sin embargo, *WallPushups* sí posee un soporte que está dentro de la media y aún así tiene una sensibilidad y una precisión igual a cero. Indicando con esto que el modelo no pudo aprender a diferenciar esta clase.

Tabla 9: Matriz de confusión con las 28 clases con menor recall del modelo recurrente en la data de validación.

	PizzaTossing	SalsaSpin	Archery	VolleyballSpiking	GolfSwing	JugglingBalls	TaiChi	Hammering	JumpingJack	Kayaking	ApplyLipstick	Shotput	PullUps	JavelinThrow	Lunges	SoccerJuggling	HighJump	BenchPress	LongJump	Rowing	BalanceBeam	CricketBowling	CricketShot	Nunchucks	BodyWeightSquats	HandstandWalking	JumpRope	WallPushups	recall	support
PizzaTossing	**22**							1																2					0.69	32
SalsaSpin	1	**20**																						2					0.67	30
Archery		1	**25**																										0.66	38
VolleyballSpiking				**23**																									0.66	35
GolfSwing					**23**							1		1															0.64	36
JugglingBalls		1				**25**						3		4										6					0.63	40
TaiChi	3						**17**									1								1					0.61	28
Hammering								**20**																		1		1	0.61	33
JumpingJack									**22**				5												4				0.59	37
Kayaking										**21**										3									0.58	36
ApplyLipstick											**15**																		0.56	27
Shotput				5								**25**		5															0.54	46
PullUps				4									**15**															1	0.54	28
JavelinThrow														**16**			3												0.52	31
Lunges						1			1						**19**	1									5				0.51	37
SoccerJuggling	1															**11**													0.50	22
HighJump				1										7			**18**		2										0.49	37
BenchPress															14			**23**											0.48	48
LongJump												4		6		6			**18**										0.46	39
Rowing																				**5**									0.45	11
BalanceBeam																					**14**								0.45	31
CricketBowling				7										4								12	3			3			0.33	36
CricketShot																						**20**	14						0.29	49
Nunchucks			1		4	1			3					1									3	7					0.21	34
BodyWeightSquats			3			2									6										5			9	0.17	30
HandstandWalking													3		1										6	3			0.09	34
JumpRope						1			1											3									0.00	5
WallPushups		4				5							7			1								1					0.00	35

En la siguiente figura se aprecian frames de quince muestras de la clase *Billiards* la cual alcanzó 100% de precisión y sensibilidad. Al observarlas se puede deducir que los videos siempre tendrán una mesa de billar que ocupe un área importante de la pantalla, es decir, las muestras son muy homogéneas, por tanto se justifica el importante nivel de precisión y sensibilidad alcanzados.

Figura 25: Captura de uno de los frames de 15 muestras de la clase *Billiards*, la cual obtuvo una precisión de 100% y una sensibilidad de 100% con un soporte de 20 muestras para validación y 76 para entrenamiento.

A continuación se aprecian frames de quince muestras de la clase *RockClimbingIndoor* la cual alcanzó un 86% de precisión y un 100% de sensibilidad. De ellas también es posible decir que poseen muestras muy homogéneas en cuanto a fondo, ángulo de cámara y patrón de movimiento. Por tanto el nivel de precisión y sensibilidad alcanzados tienen una mejor explicación.

Figura 26: Captura de uno de los frames de 15 muestras de la clase *RockClimbingIndoor*, la cual obtuvo una precisión de 86% y una sensibilidad de 100% con un soporte de 6 muestras para validación y 25 para entrenamiento.

Por otro lado, el siguiente grupo de frames son tomados de quince muestras de la clase *BodyWeightSquats,* la cual alcanzó un 21% de precisión y un 17% de sensibilidad. De estas imágenes, se puede apreciar que existe una mayor variabilidad en las muestras de la clase en función al ángulo de la toma, fondo de la escena y patrón de movimiento realizado. Este análisis visual simple puede arrojar algunas luces sobre las métricas alcanzadas, considerando que esta clase tenía una cantidad de muestras de 112 muy cercana a la media global de 131.88.

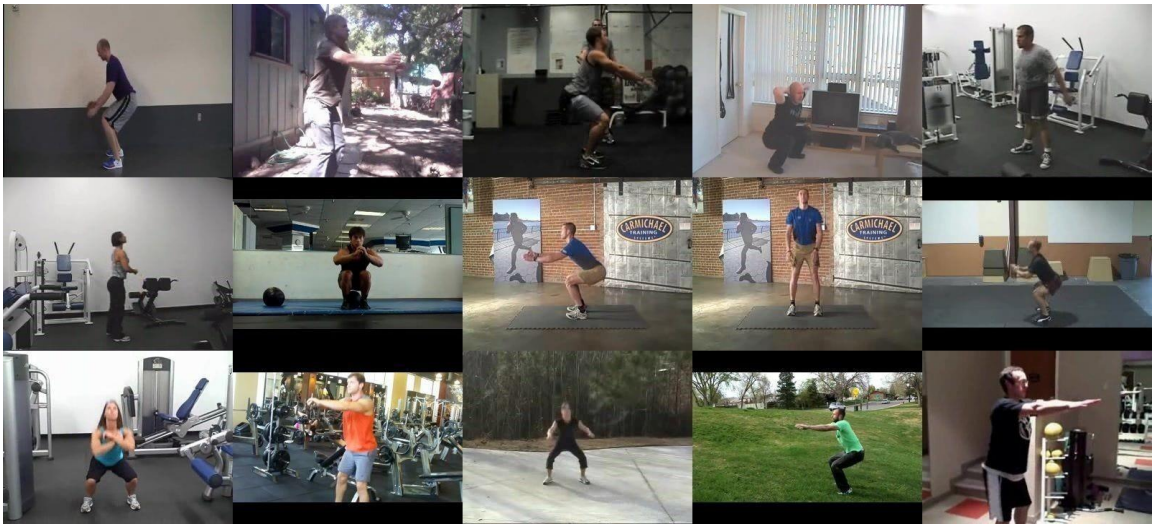

Figura 27: Captura de uno de los frames de 15 muestras de la clase *BodyWeightSquats*, la cual obtuvo una precisión de 21% y una sensibilidad de 17% con un soporte de 30 muestras para validación y 82 para entrenamiento.

En cambio, la clase *WallPushups,* la cual alcanzó un 0% de precisión y de sensibilidad con un soporte de 130 muestras, dejando claro que el modelo no es capaz de discriminar esta clase. Haciendo un análisis visual, se aprecia que existe una gran variabilidad en función al ángulo de la toma y fondo de la escena. Además muchos de los frames incluyen grandes porciones de franjas negras.

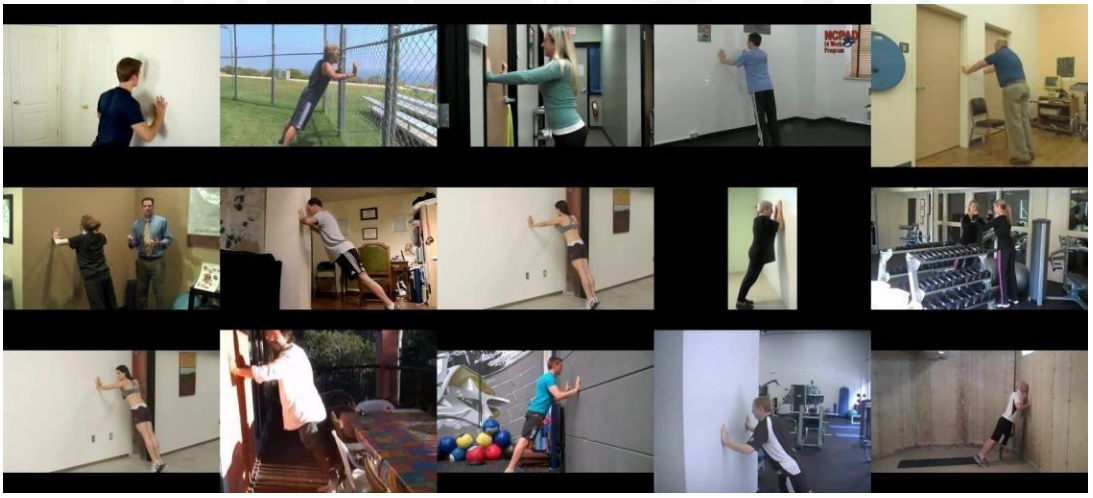

Figura 28: Captura de uno de los frames de 15 muestras de la clase *WallPushups*, la cual obtuvo una precisión de 0% y una sensibilidad de 0% con un soporte de 35 muestras para validación y 95 para entrenamiento.

En consecuencia, se puede decir que, en las clases con menor precisión alcanzada se tiene como patrón común una mayor variabilidad en la muestra y mayor ruido debido al ángulo de la toma y al fondo de la escena. Es decir que, el modelo de extracción de características no es capaz de diferenciar el fondo de la escena de la acción en cuestión.

Por último, de forma complementaria, se realizó la predicción con un ejemplar del conjunto de datos de validación con la finalidad de observar los valores del vector de salida de la red. En la Figura 29 se puede apreciar la predicción del modelo para una muestra de video de la categoría *BlowingCandles*, en ella se aprecia que el máximo valor obtenido del vector es 0.46329954, que señala la clase correcta.

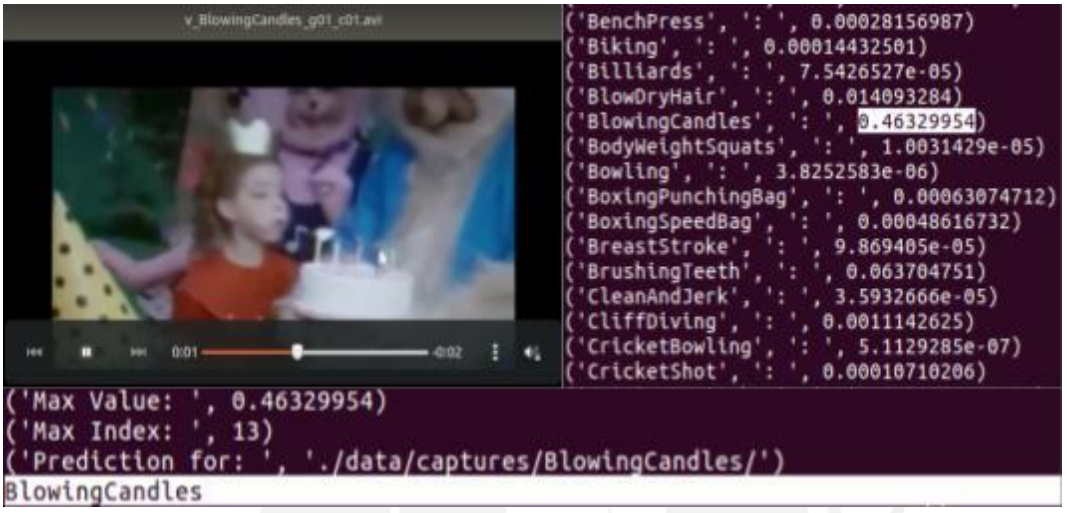

Figura 29: Predicción del modelo para una muestra de acción de la categoría *BlowingCandles*.

5.3.2. Enfoque de red convolucional y LSTM sobre VIRAT 2.0

Debido a que el objetivo de la investigación es el soporte de la tarea de videovigilancia, se optó por utilizar el dataset VIRAT 2.0 Ground, el cual contiene muestras capturadas de cámaras de seguridad. Además, cabe resaltar que éstas tomas son de plano abierto, es decir, no hacen énfasis o acercamiento en la acción en sí misma sino en la escena completa.

5.3.2.1. Preprocesamiento de datos

El preprocesamiento de los datos es muy similar al de UCF101. Primero, se distribuyen los archivos de video en tantas carpetas como acciones se tiene. Luego, se extraen los frames de cada uno de los videos y se distribuyen como archivos dentro de subcarpetas con el nombre del archivo de video en cuestión alojadas en las carpetas con el nombre de la acción correspondiente. Esto se realiza de forma automatizada con la librería *ffmpeg* y se repite el proceso de extracción de frames sin perder la estructura de directorios y subdirectorios. Además, se debe considerar que los videos

están codificados a una tasa de 30 frames por segundo en su mayoría.

Seguidamente, se distribuye aleatoriamente las muestras entre el conjunto de entrenamiento y el conjunto de validación. El ratio de distribución en este caso fue de 75% de muestras para entrenamiento y 25% para validación, tomando en consideración que este dataset tiene un volumen considerablemente menor al del UCF101 y que, en consecuencia, algunas clases disponen de muy pocas muestras para realizar el entrenamiento tal como se aprecia en la Tabla 10 las acciones *LoadingObjectToVehicle* y *Running* poseen poco más de 20 muestras totales. Agregado a lo anterior, los datos se encuentran desbalanceados: con una media de muestras totales por acción de 141.36 y una desviación estándar de 230.23 con un mínimo de 21 y un máximo de 822 muestras. Es decir, existe un enorme desbalance en los datos y por tanto una gran variabilidad en la cantidad de muestras por acción.

Tabla 10: Distribución de la cantidad de muestras por acción para entrenamiento y validación en un ratio de 75% y 25% alcanzando un mínimo de 21 y un máximo de 822 videos por acción humana. La acción *Digging* no posee muestras.

Action	train	test	total
LoadingObjectToVehicle	16	5	21
UnloadingObjectFromVehicle	45	14	59
OpeningVehicleCarTrunk	32	10	42
ClosingVehicleCarTrunk	31	10	41
GettingIntoVehicle	84	27	111
GettingOutOfVehicle	73	24	97
Gesturing	39	12	51
CarryingObject	617	205	822
Running	17	5	22
EnteringFacility	117	39	156
ExitingFacility	100	33	133
TOTAL	**1171**	**384**	**1555**

De forma seguida, se realiza la reducción de muestras en función a la longitud, medida en frames, del video. El filtrado se realiza descartando los videos con una longitud menor a 40 o mayor a 300 frames. Luego de dicha reducción, la variabilidad de la cantidad de muestras por acción disminuye notablemente, tal como se muestra en la Tabla 11. Sin embargo, el desbalance aún persiste aunque en menor grado. Esto se puede apreciar al observar las tres clases con menor cantidad de muestras: *LoadingObjectToVehicle* (19)*, Running* (19)*, Gesturing* (38) y las tres clases con mayor cantidad de muestras: *ExitingFacility* (133), *EnteringFacility* (154) y *CarryingObject*

(243). Por último, se genera el archivo CSV con la misma estructura que el archivo CSV del dataset UCF101.

Tabla 11: Distribución de la cantidad de muestras por acción antes y después de filtrar las acciones por cantidad de frames.

Action	total	total filtered
LoadingObjectToVehicle	21	19
UnloadingObjectFromVehicle	59	58
OpeningVehicleCarTrunk	42	42
ClosingVehicleCarTrunk	41	40
GettingIntoVehicle	111	96
GettingOutOfVehicle	97	90
Gesturing	51	38
CarryingObject	822	243
Running	22	19
EnteringFacility	156	154
ExitingFacility	133	133
TOTAL	**1555**	**932**
Average	141.36	84.73
Standard deviation	230.23	69.06
Mín	21	19
Max	822	243

5.3.2.2. Procesamiento de datos

En esta fase del proceso general, se utiliza el mismo modelo basado en dos arquitecturas, una para la extracción de características por frame y la otra para la clasificación utilizando una red LSTM. Teniendo esta consideración, el resto de pasos es muy similar a los utilizados para el dataset UCF101.

En primer lugar, se reduce la cantidad de frames de todas las muestras de forma que se tenga un número idéntico de frames por muestra, en este caso, 40 frames. Esta reducción se consigue al extraer los j-ésimos frames de la lista de entrada, donde la posición j es, a lo más, múltiplo de siete. Luego, se procede con la extracción de características de cada uno de los frames utilizando la red Inception V3, pero tomando el vector de salida de la última capa *GlobalAveragePooling2.* Luego de ello, se concatena los 40 vectores extraídos y se almacenan en un archivo por cada muestra de video.

De forma seguida, se procede con el entrenamiento de la segunda red LSTM. Para ello, se carga cada uno de los vectores serializados en archivos, en una lista que contiene 40 vectores (uno por cada frame de video) los cuales poseen una longitud de 2048 (que es el vector obtenido de la capa *Global Average Polling* de la red Inception-V3). La salida obtenida de la capa LSTM se procesa usando una capa *Flatten* de modo que se obtiene un vector de una dimensión y de longitud 81920. Finalmente, para la clasificación, se introduce este vector en una capa completamente conectada, *Dense*, de 512 unidades y luego en otra capa *Dense* con 11 unidades con una función de activación Softmax para la clasificación. En la Figura 30 se aprecia la arquitectura usada, la cual es idéntica a la aplicada para el dataset UCF101, con la diferencia de que este subconjunto de VIRAT 2.0 posee solo 11 clases. En esta red la capa LSTM posee una salida de la forma (None, 40, 2048) y un total de 33,562,624 parámetros, la capa Flatten posee una salida de la forma (None, 81920), la primera capa Dense posee una salida de la forma (None, 512) y un total de 41,943,552 con Dropout, y la última capa Dense posee una salida de la forma (None, 11) con un total de 5,643 parámetros. Esta red tiene un total de 75,511,819 parámetros entrenables.

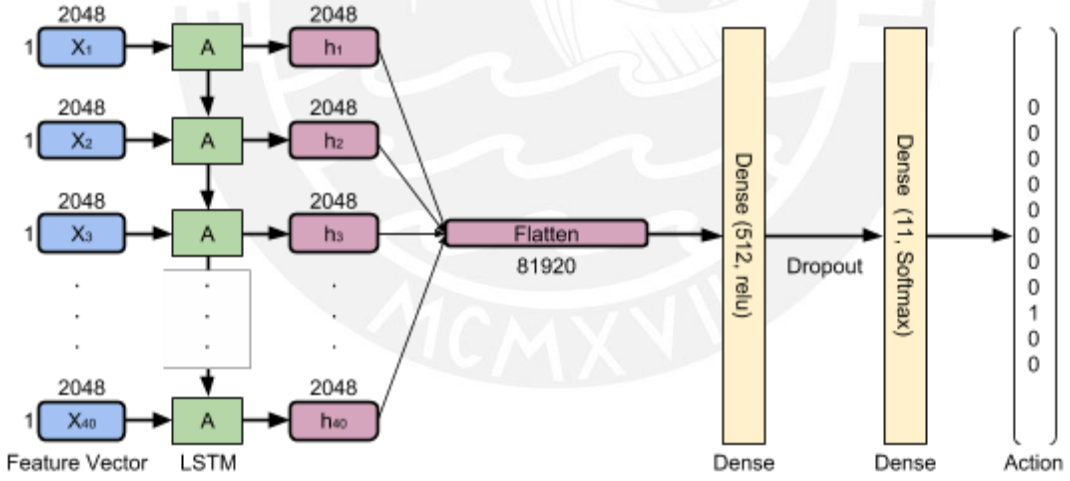

Figura 30: Arquitectura de la red con memoria para la clasificación de las 11 acciones del *dataset* VIRAT 2.0 Ground.

5.3.2.3. Resultados

Luego de introducir el dataset VIRAT 2.0 Ground (con 663 muestras para entrenamiento y 270 para validación) a la arquitectura antes descrita, se obtuvo los siguientes resultados: El entrenamiento de la red LSTM, utilizando el dataset en cuestión, y la validación tardaron en conjunto un total de 3319.6 segundos, es decir,

55.3 minutos.

Además, se configuró el algoritmo para que realizara 1000 épocas con un *batch size* de 128, adicionalmente se utilizó un optimizador Adam (Kingma & Ba, 2014) con un learning rate de 0,002478752 (es decir de e^{-6}) y una función de pérdida de tipo entropía cruzada categórica. Con los parámetros mencionados anteriormente se obtienen los siguientes resultados al finalizar el entrenamiento y validación:

- Para los datos de entrenamiento:
  - loss function: 1.1600
  - categorical accuracy: 0.5309
  - top 5 categorical accuracy: 0.9819
- Para los datos de validación:
  - loss function: 1.6099
  - categorical accuracy: 0.4222
  - top 5 categorical accuracy: 0.9111

De estos valores, en primer lugar, se aprecia que la función de pérdida en los datos de entrenamiento, a diferencia que con UCF101, no alcanza un valor muy bajo, es decir, que el modelo no es tan capaz de aprender en este conjunto de datos. Esto se corrobora al visualizar la figura que se muestra a continuación, donde se observa la gráfica de la función de pérdida en los datos de entrenamiento de color azul y en los datos de validación de color anaranjado. Además, se puede observar que la función en los datos de validación comienza a perder capacidad de generalización a partir de la época 500 y por tanto la función de pérdida comienza a incrementar su valor y el modelo se sobre ajusta.

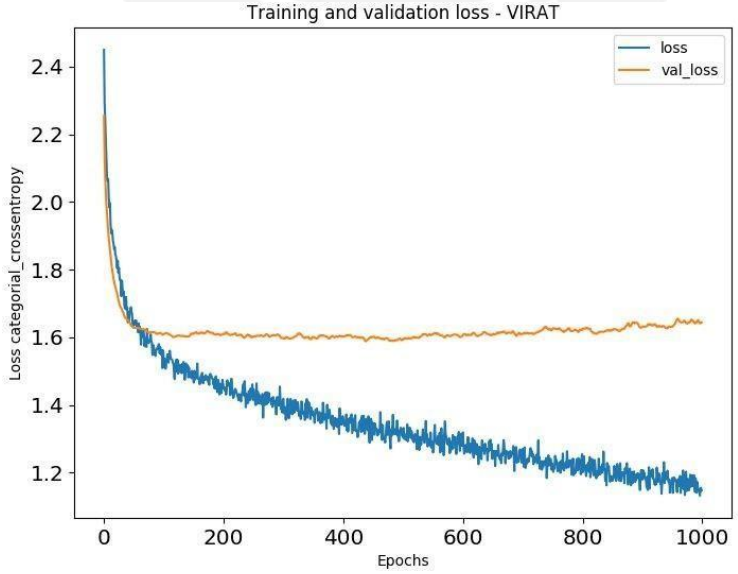

Figura 31: Gráfica de la función de pérdida para los datos de entrenamiento (de color azul) y

para los datos de validación (de color anaranjado) del modelo de red con memoria para el conjunto de datos VIRAT 2.0 Ground.

En segundo lugar, se puede apreciar que el accuracy categórico en los datos de entrenamiento es de 53.09% mientras que en lo datos de validación alcanza el valor de 42.22%. Además, se deduce que el modelo comienza a sobreentrenarse o hacer overfitting a partir de la época 500. Adicionalmente, la tasa de acierto categórico top-5 alcanza el valor de 98.19% para los datos de entrenamiento y de 91.11% para los datos de validación lo cual se aprecia en la Figura 32. Estos valores revelan que existe mayor dificultad para poder realizar la tarea de clasificación de acciones en este conjunto de datos.

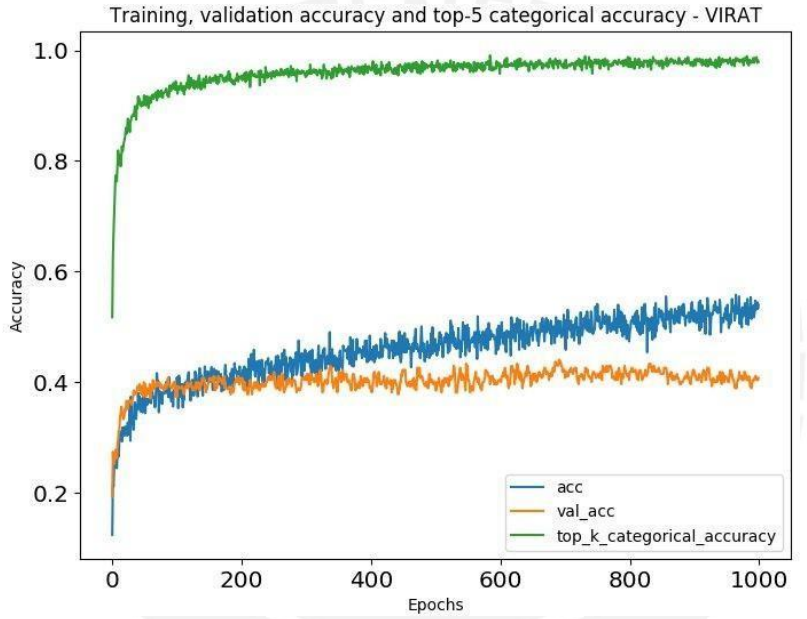

Figura 32: Gráfica de la precisión categórica para los datos de entrenamiento (de color azul) y para los datos de validación (de color anaranjado) del modelo de red con memoria para el conjunto de datos VIRAT 2.0 Ground. Además se aprecia la tasa de acierto categórico top-5.

Por otro lado, el modelo tiene una precisión promedio de 42%, siendo las clases *CarryingObject*, *EnteringFacility*, *ExitingFacility* las mejor clasificadas con valores de 61%, 54% y 52% respectivamente. De estas *EnteringFacility* y *CarryingObject* tuvieron sensibilidades de 67% y de 61% con un soporte de 39 y 57 muestras respectivamente. Contrario a lo anterior, las clases *ClosingVehicleCarTrunk*, *OpeningVehicleCarTrunk*, *UnloadingObjectFromVehicle* tuvieron la menor precision, siendo en los dos primeros casos igual a cero tal como se aprecia en la Table 12.

Tabla 12: Métricas obtenidas por clase, para el conjunto de datos VIRAT 2.0 Ground: *precision*, sensibilidad (*recall*) y la utilidad teórica (F1-Score) además del soporte para cada clase en el conjunto de datos de validación.

action	precision	recall	f1-score	support
LoadingObjectToVehicle	0.25	0.2	0.22	5
Running	0.25	0.25	0.25	4
EnteringFacility	**0.54**	**0.67**	0.6	39
ExitingFacility	**0.52**	0.36	0.43	33
UnloadingObjectFromVehicle	0.2	0.23	0.21	13
OpeningVehicleCarTrunk	0	0	0	10
ClosingVehicleCarTrunk	0	0	0	10
GettingIntoVehicle	0.31	0.36	0.33	25
GettingOutOfVehicle	0.32	0.35	0.33	23
Gesturing	0.36	0.5	0.42	8
CarryingObject	**0.61**	**0.61**	0.61	57
avg / total	**0.42**	**0.44**	**0.43**	**227**

De forma complementaria, se presenta la matriz de confusión la cual evidencia la dificultad del modelo para diferenciar acciones en secuencias de videovigilancia. Además, muestra un fenómeno particular relacionado con las clases que involucran automóviles: al modelo le resulta muy difícil diferenciar entre ellas. Finalmente, muchos de los eventos son confundidos con la acción *CarryingObject*.

Tabla 13: Matriz de confusión con las 11 clases de acción del modelo VIRAT 2.0 Ground, indicando el recall del y soporte de cada clase.

True \ Predicted Class \ Class	LoadingObjectToVehicle	Running	EnteringFacility	ExitingFacility	UnloadingObjectFromVehicle	OpeningVehicleCarTrunk	ClosingVehicleCarTrunk	GettingIntoVehicle	GettingOutOfVehicle	Gesturing	CarryingObject	recall	support
LoadingObjectToVehicle	1	1			1		1	1				0.20	5
Running		1		1							2	0.25	4
EnteringFacility			26	6						2	5	0.67	39
ExitingFacility			14	12						2	5	0.36	33
UnloadingObjectFromVehicle					3	1	1	5	2		1	0.23	13
OpeningVehicleCarTrunk	1				1		1	3	4			0.00	10
ClosingVehicleCarTrunk					3			7				0.00	10
GettingIntoVehicle						1	3	9	10		2	0.36	25
GettingOutOfVehicle	1				4	2		3	8	2	3	0.35	23
Gesturing										4	4	0.50	8
CarryingObject	1	2	8	4	3	1		1	1	1	35	0.61	57

Para finalizar este análisis, se muestra a continuación frames de diferentes acciones: *GettingIntoVehicle, LoadingObjectToVehicle* de 31% y 25% de precisión y también se muestra un frame de la acción *CarryingObject* la cual obtuvo un 61% de precisión. Además, se puede apreciar que, utilizando solamente los frames, resulta difícil para un humano diferenciar la acción debido a que estas se desarrollan en áreas muy pequeñas de la pantalla. Es necesario agregar la información temporal del movimiento usando, en este caso, la propiedad de memoria de la red LSTM.

Figura 33: Frame de una muestra de la acción *GettingIntoVehicle*. La acción está enfatizada con un rectángulo rojo para su mejor diferenciación.

Figura 34: Frame de una muestra de la acción *LoadingObjectToVehicle*. La acción está enfatizada con un rectángulo rojo para su mejor diferenciación.

Figura 35: Frame de una muestra de la acción *CarryingObject*. La acción está enfatizada con un rectángulo rojo para su mejor diferenciación.

5.3.3. Enfoque de red convolucional y LSTM sobre VIRAT 2.0 con los frames de las acciones recortadas

Una de las limitaciones observadas en el experimento anterior fue que, debido a la apertura de las tomas, el área del frame que contiene información importante queda relegada a un espacio muy pequeño, introduciendo información innecesaria en los vectores de características extraídos por la red convolucional Inception V3. Debido a ello, se optó por introducir al modelo, únicamente el área donde se desarrolla la acción, luego de recortarla. Esto se pudo realizar dado que el dataset VIRAT 2.0 Ground contiene la información del box donde se desarrolla la acción por cada frame.

5.3.3.1. Preprocesamiento de datos

El preprocesamiento de la data fue exactamente igual al experimento anterior, con la salvedad que se tuvo que añadir un script para la extracción o *cropping* del box donde se desarrolla la acción por cada frame.

Luego, se realizó la distribución aleatoria de las muestras en una proporción de 75% para entrenamiento y 25% para validación. Por tanto, la distribución de muestras por acción terminó siendo la misma que para el VIRAT de frames completos. Finalmente, se realizó el filtrado de las muestras por acción de acuerdo a la longitud de frames.

5.3.3.2. Procesamiento de datos

Esta fase del procesamiento es la misma que la utilizada en el anterior experimento. Primero se estandariza la cantidad de frames por muestra a 40. Luego, se procede con la extracción de características de cada uno de los frames utilizando la red Inception V3 de la última capa *GlobalAveragePooling2* y se almacenan las características en un archivo de texto por cada video. Finalmente, se procede con la clasificación utilizando la misma arquitectura que en el experimento anterior.

5.3.3.3. Resultados

El entrenamiento de la red LSTM, utilizando el dataset en cuestión, y la validación tardaron en conjunto un total de 3694.8 segundos, es decir, 1 hora y 1.5 minutos.

Además, se configuró el algoritmo para que realizara 1000 épocas con un *batch size* de 128, adicionalmente se utilizó un optimizador Adam con un learning rate de 0,002478752 (es decir de e^{-6}) y una función de pérdida de tipo entropía cruzada categórica. Con los parámetros mencionados anteriormente se obtienen los siguientes resultados al finalizar el entrenamiento y validación:

- Para los datos de entrenamiento:
    - loss function: 0.1868
    - categorical accuracy: 0.9460
    - top 5 categorical accuracy: 1.0000
- Para los datos de validación:
    - loss function: 1.5483
    - categorical accuracy: 0.5965
    - top 5 categorical accuracy: 0.9474

Estos valores denotan una mejora con respecto al experimento anterior. El modelo además muestra que es capaz de aprender dentro del conjunto de datos. Esto se corrobora al visualizar la figura que se muestra a continuación, donde se observa la gráfica de la función de pérdida en los datos de entrenamiento de color azul y en los datos de validación de color anaranjado. Además, se puede observar que la función en los datos de validación comienza a perder capacidad de generalización a partir de la época 300 y por tanto la función de pérdida comienza a incrementar su valor y el modelo se sobre ajusta.

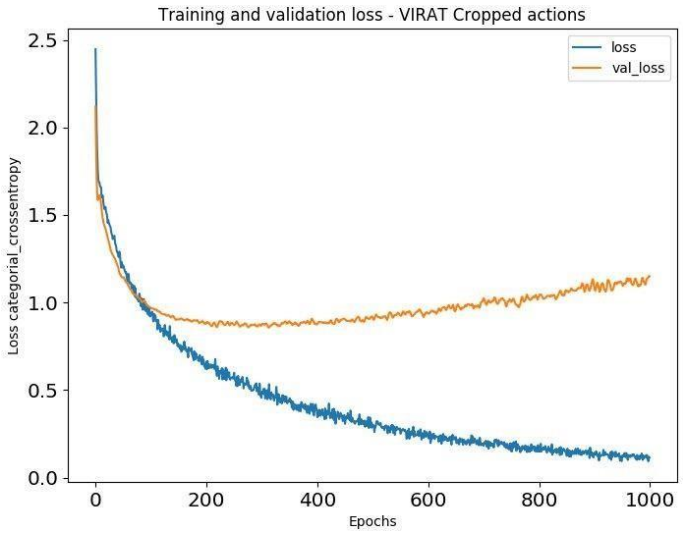

Figura 36: Gráfica de la función de pérdida para los datos de entrenamiento (de color azul) y para los datos de validación (de color anaranjado) del modelo de red con memoria para el conjunto de datos VIRAT 2.0 Ground de acciones recortadas.

En segundo lugar, se puede apreciar que el accuracy categórico en los datos de entrenamiento es de 94.60% mientras que en lo datos de validación alcanza el valor de 59.65%. Además, se deduce que el modelo comienza a sobreentrenarse o hacer overfitting a partir de la época 300. Adicionalmente, la tasa de acierto categórico top-5 alcanza el valor de 100% para los datos de entrenamiento y de 94.74% para los datos de validación lo cual se aprecia en la Figura 37. Estos valores revelan que existe mayor dificultad para poder realizar la tarea de clasificación de acciones en este conjunto de datos.

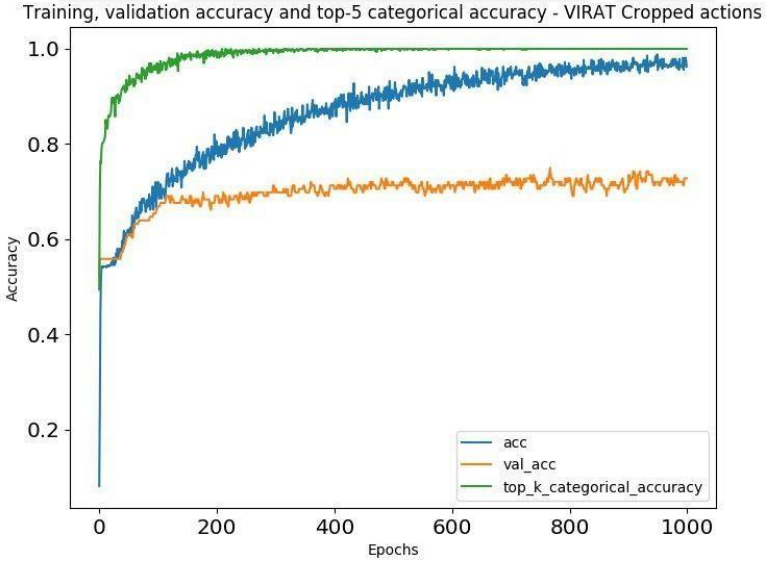

Figura 37: Gráfica de la precisión categórica para los datos de entrenamiento (de color azul) y para los datos de validación (de color anaranjado) del modelo de red con memoria para el conjunto de datos VIRAT 2.0 Ground recortada. Además se aprecia la tasa de acierto categórico top-5.

Por otro lado, el modelo tiene una precisión promedio de 55%, la cual difiere del accuracy categórico (línea de color anaranjado en la Figura 37) debido a la fórmula utilizada para calcularla (ver sección Categorical accuracy y sección Precisión) siendo las clases *EnteringFacility*, *ExitingFacility* y *CarryingObject* las mejor clasificadas con valores de 77%, 73% y 68% respectivamente. De estas, las dos primeras, tuvieron sensibilidades de 77% y de 73% (que coincidieron con los valores de la precision) con un soporte de 39 y 33 muestras respectivamente y *CarryingObject* tuvo una sensibilidad de 95% con un soporte de 61 muestras. Contrario a lo anterior, las clases *LoadingObjectToVehicle, Running y OpeningVehicleCarTrunk* tuvieron la menor precisión, siendo en todos los casos igual a cero tal como se aprecia en la tabla siguiente.

Tabla 14: Métricas obtenidas por clase, para el conjunto de datos VIRAT 2.0 Ground: *precision*, sensibilidad (*recall*) y la utilidad teórica (F1-Score) además del soporte para cada clase en el conjunto de datos de validación.

action	precision	recall	f1-score	support
LoadingObjectToVehicle	0	0	0	5
Running	0	0	0	3
EnteringFacility	0.77	0.77	0.77	39
ExitingFacility	0.73	0.73	0.73	33
UnloadingObjectFromVehicle	0.25	0.21	0.23	14
OpeningVehicleCarTrunk	0	0	0	10
ClosingVehicleCarTrunk	0.2	0.1	0.13	10
GettingIntoVehicle	0.31	0.35	0.33	23
GettingOutOfVehicle	0.41	0.39	0.4	23
Gesturing	1	0.43	0.6	7
CarryingObject	0.68	0.95	0.79	61
avg / total	**0.55**	**0.6**	**0.56**	**228**

De forma complementaria, se presenta la matriz de confusión. Se observa que disminuye la incapacidad de diferenciar entre clases que involucren automóviles. Finalmente, muchos de los eventos siguen siendo confundidos con la acción *CarryingObject*.

Tabla 15: Matriz de confusión con las 11 clases de acción del modelo VIRAT 2.0 Ground, indicando el recall del y soporte de cada clase.

True \ Predicted Class \ Class	LoadingObjectToVehicle	Running	EnteringFacility	ExitingFacility	UnloadingObjectFromVehicle	OpeningVehicleCarTrunk	ClosingVehicleCarTrunk	GettingIntoVehicle	GettingOutOfVehicle	Gesturing	CarryingObject	recall	support
LoadingObjectToVehicle					1			2			2	0.00	5
Running											3	0.00	3
EnteringFacility			30	7							2	0.77	39
ExitingFacility			8	24							1	0.73	33
UnloadingObjectFromVehicle					3	1	4		2		4	0.21	14
OpeningVehicleCarTrunk					1			3	3		3	0.00	10
ClosingVehicleCarTrunk					1		1	1	1		6	0.10	10
GettingIntoVehicle			1		4	1		8	7		2	0.35	23
GettingOutOfVehicle					2	1		11	9			0.39	23
Gesturing										3	4	0.43	7
CarryingObject				2				1			58	0.95	61

En las Figuras 38 y 39 se aprecia la diferencia entre un video de frame completo y uno de acción recortada. De esta forma se elimina gran cantidad de información contextual que puede resultar innecesaria de cada frame.

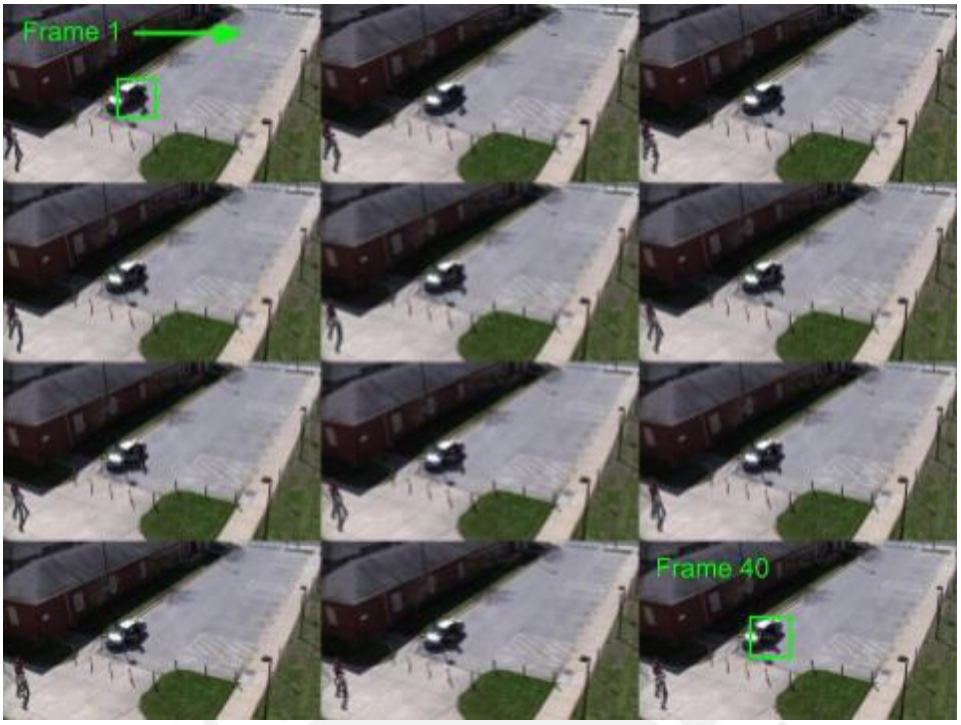

Figura 38: Secuencia representativa de frames de una muestra de la acción *GettingIntoVehicle*.

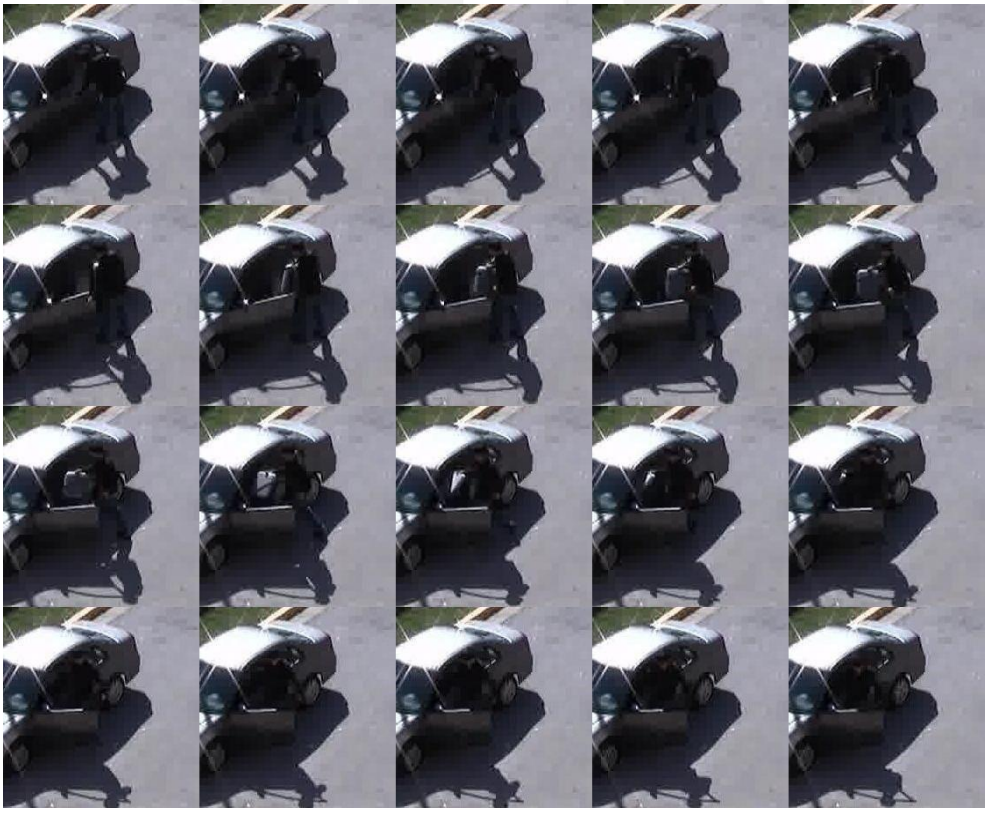

Figura 39: Secuencia de frames recortada de una muestra de la acción *GettingIntoVehicle*.

Finalmente, una de las dificultades más grandes encontradas fue la simultaneidad de desarrollo de acciones. Es decir, se tenían muestras de más de una acción desarrollándose en un mismo instante. Esto era muy difícil de poder diferenciar con el enfoque anterior.

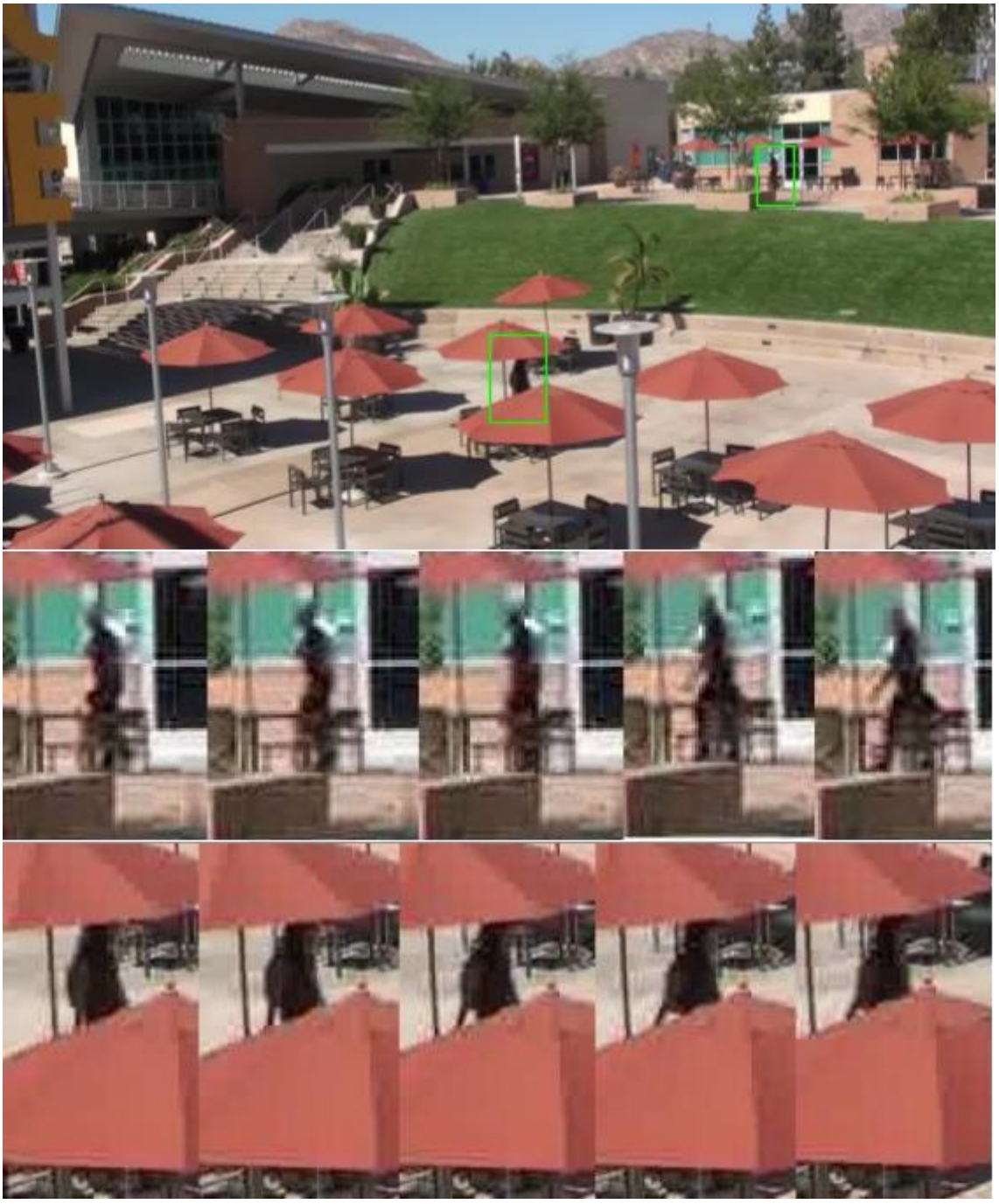

Figura 40: Dos acciones desarrolladas en una misma secuencia de frames. En la primera imagen se aprecia un frame representativo de la secuencia de plano abierto y en las siguientes imágenes, se muestran frames consecutivos de las dos acciones desarrollándose en simultáneo.

5.4. Comparación de resultados

En general, el enfoque de recorte de la acción, produjo mejores resultados. En la precisión categórica se obtuvo una mejora de cerca de 20 puntos porcentuales y en la tasa de acierto categórico top-5 de 4 puntos porcentuales. Esto ratifica la importancia de no introducir el frame completo en el caso de identificación de acciones en video. Como se aprecia en la Tabla 16, para frames enteros se obtuvo 1.16 de pérdida con el conjunto de entrenamiento y 1.6099 con el de validación.

Tabla 16: Métricas generales del modelo para datasets de VIRAT 2.0 Ground

	Entire frame	**Cropped frame**
	Training	**Training**
loss function	1.16	0.1868
categorical accuracy	0.5309	0.946
top 5 categorical accuracy	0.9819	1
	Validation	**Validation**
loss function	1.6099	1.5483
categorical accuracy	0.4222	0.5965
top 5 categorical accuracy	0.9111	0.9474

Por otro lado, la precisión y sensibilidad por clase se incrementaron notablemente. De forma particular en las tres clases con mejores métricas. Como se observa en la Tabla 17, los frames cortados presentan mejores resultados, siendo que con las clases EnteringFacility y ExitingFacility se obtienen 0.77 y 0.73 de F1-Score, respectivamente. Cabe resaltar que las clases *LoadingObjectToVehicle* y *Running* no presentaron ninguna mejora, por el contrario, la precisión disminuyó a 0. Las muestras del conjunto de validación de la primera acción se confundieron con las acciones *UnloadingObjectFromVehicle, GettingIntoVehicle y CarryingObject* ver Tabla 15. Las muestras de la segunda acción se confundieron con la acción *CarryingObject.* En principio, se sospecha que estas acciones fueron afectadas debido a la poca cantidad de muestras para entrenamiento (son las que menos muestras tienen) pero también es posible que se haya perdido un poco de información contextual en esas muestras en específico.

Tabla 17: Comparación de las métricas obtenidas por clase, para el conjunto de datos VIRAT 2.0 Ground de frame completo y de frame recortado para enfatizar la acción.

	Entire frame	Cropped frame	Entire frame	Cropped frame	Entire frame	Cropped frame	
	precision	**precision**	**recall**	**recall**	**f1-score**	**f1-score**	**support**
LoadingObjectToVehicle	0.25	0	0.2	0	0.22	0	5
Running	0.25	0	0.25	0	0.25	0	4
EnteringFacility	**0.54**	**0.77**	**0.67**	**0.77**	0.6	0.77	39
ExitingFacility	**0.52**	**0.73**	**0.36**	**0.73**	0.43	0.73	33
UnloadingObjectFromVehicle	0.2	0.25	0.23	0.21	0.21	0.23	14
OpeningVehicleCarTrunk	0	0	0	0	0	0	10
ClosingVehicleCarTrunk	0	0.2	0	0.1	0	0.13	10
GettingIntoVehicle	0.31	0.31	0.36	0.35	0.33	0.33	24
GettingOutOfVehicle	0.32	0.41	0.35	0.39	0.33	0.4	23
Gesturing	0.36	1	0.5	0.43	0.42	0.6	8
CarryingObject	**0.61**	**0.68**	**0.61**	**0.95**	0.61	0.79	59
avg / total	**0.42**	**0.55**	**0.44**	**0.6**	**0.43**	**0.56**	**228**

6. Conclusiones y Trabajo Futuro

La naturaleza espacio-temporal de las acciones en video requieren métodos que sean capaces de capturar características de los frames de video y de la transición temporal entre frames. Sin embargo, las convoluciones o las redes recurrentes no son suficientes para lograr una extracción de características óptima. Otros de los métodos vistos en el estado del arte hacen uso de la fusión de características obtenidas utilizando técnicas dentro del marco de redes neuronales (CNN, LSTM, 3D CNN, etc.) pero también incluyen técnicas que han mostrado ser muy efectivas antes del uso intensivo del *deep learning* tales como flujo óptico de movimiento (obtenido a partir de dos frames consecutivos tal como se muestra en la Figura 41), descriptores de movimiento, etc. Todo esto con la finalidad de incrementar la dimensionalidad de las características extraídas de cada video que es lo que enriquece y al mismo tiempo diferencia una acción contenida en una secuencia de frames de un objeto contenido en una imagen.

Figura 41: Frame de flujo óptico denso de movimiento obtenido a partir de dos frames de entrada consecutivos.

Otras técnicas necesarias para incrementar el rendimiento general del modelo tiene que ver con el tratamiento de los datos. En principio, el balanceo de los datos a partir del uso de técnicas de aumentación de datos que permitan generar variaciones aleatorias aplicadas de forma común a todos los frames de una acción. De esta forma se puede realizar la inversión horizontal, modificación de los canales R, G o B, desplazamiento horizontal o vertical de los frames recortados (siempre que el marco no se encuentre en uno de los bordes del frame general de forma que se tenga que rellenar con píxeles negros) y variaciones en el zoom. Todas estas técnicas combinadas permitirían aumentar considerablemente la cantidad de datos de la muestra y además permitiría homogeneizar los datos.

Finalmente, para el uso de la solución en un sistema de videovigilancia se debería implementar una técnica de segmentación automática de acciones que permita recortar automáticamente las múltiples acciones que se pueden identificar en una secuencia de videovigilancia.

7. Bibliografía

Aggarwal, J. K., & Ryoo, M. S. (2011). Human activity analysis. *ACM Computing Surveys (CSUR), 43*(3), 1-43. doi:10.1145/1922649.1922653

Ali, & Senan. (2016). A review on violence video classification using convolutional neural networks. *International Conference on Soft Computing and Data Mining,* Cham. pp. 130-140.

Bellard, F. (2005). *Ffmpeg multimedia system.* Retrieved November, 2017, from https://www.ffmpeg.org/about.html

Carreira, J., & Zisserman, A. (2017). Quo vadis, action recognition? A new model and the kinetics dataset. *arXiv Preprint arXiv:1705.07750,* , 4724-4733. doi:1705.07750

Chen, C., & Xia, L. (2015). In - (Ed.), *Recurrent neural network and long short-term memory* (- ed.). Ohio: Russ College of Engineering and Technology.

Cheng, G., Wan, Y., Saudagar, A. N., Namuduri, K., & Buckles, B. P. (2015). Advances in human action recognition: A survey. Retrieved from http://arxiv.org/abs/1501.05964

Chunhui Ding, Shouke Fan, Ming Zhu, Weiguo Feng, & Baozhi Jia. (2014). Violence detection in video by using 3D convolutional neural networks. *International Symposium on Visual Computing,* Cham. pp. 551-558.

Dai, Q., Zhao, R., Wu, Z., Wang, X., Gu, Z., Wu, W., et al. (2015). Fudan-huawei at MediaEval 2015: Detecting violent scenes and affective impact in movies with deep learning. *MediaEval,* pp. 1-3.

De Souza, F. D., Chavez, G. C., do Valle Jr, Eduardo A, & Araújo, A. d. A. (2010). Violence detection in video using spatio-temporal features. Paper presented at the *Graphics, Patterns and Images (SIBGRAPI), 2010 23rd SIBGRAPI Conference On,*

pp. 224-230.

Deniz, O., Serrano, I., Bueno, G., & Kim, T. (2014). Fast violence detection in video. Paper presented at the *Computer Vision Theory and Applications (VISAPP), 2014 International Conference On, , 2.* pp. 478-485.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735-1780. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.676.4320&rep=rep1&type=pdf

Hu, Y., Cao, L., Lv, F., Yan, S., Gong, Y., & Huang, T. S. (2009). Action detection in complex scenes with spatial and temporal ambiguities. Paper presented at the Retrieved from http://scholarbank.nus.edu.sg/handle/10635/69153

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning,* pp. 448-456.

Jain, A. K., Jianchang Mao, & Mohiuddin, K. M. (1996). Artificial neural networks: A tutorial. *Computer, 29*(3), 31-44. doi:10.1109/2.485891

Jain, A. K., Mao, J., & Mohiuddin, K. M. (1996). Artificial neural networks: A tutorial. *Computer, 29*(3), 31-44.

Joint Committee for Guides in Metrology. (2008). In JCGM/WG 2 (Ed.), *International vocabulary of metrology - basic and general concepts and associated terms (VIM)* (3rd ed.). France: Bureau International des Poids et Mesures.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Li Fei-Fei. (Jun 2014). Large-scale video classification with convolutional neural networks. Paper presented at the *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* Stanford. pp. 1725-1732. doi:10.1109/CVPR.2014.223

Kawaguchi, K. (2000). A multithreaded software model for backpropagation neural

network applications. The University of Texas at El Paso).

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv Preprint arXiv:1412.6980,*

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Paper presented at the *Advances in Neural Information Processing Systems,* pp. 1097-1105.

Lan, Z., Zhu, Y., & Hauptmann, A. G. (2017). Deep local video feature for action recognition. *Computer Vision and Pattern Recognition Workshops (CVPR),* , 1219-1225. Retrieved from http://arxiv.org/abs/1701.07368

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics, 5*(4), 115-133.

Mikolov, T., Karafit, M., Burget, L., Cernock, J., & Khudanpur, S. (2010). Recurrent neural network based language model. *Interspeech, , 2.* pp. 3.

Nievas, E. B., Suarez, O. D., García, G. B., & Sukthankar, R. (2011). Violence detection in video using computer vision techniques. Paper presented at the *International Conference on Computer Analysis of Images and Patterns,* pp. 332-339.

Oh, S., Hoogs, A., Perera, A., Cuntoor, N., Chen, C., Lee, J. T., et al. (2011). A large-scale benchmark dataset for event recognition in surveillance video. Paper presented at the *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference On,* pp. 3153-3160.

Organización Mundial de la Salud. (2012). *Violencia.* Retrieved 5/05/, 2017, from http://www.who.int/topics/violence/es/

Peemen, M. (2013). *Convolutional neural network (CNN).* Retrieved 20/07/, 2017, from https://sites.google.com/site/5kk73gpu2013/assignment/cnn

Rojas, R. (1996). The backpropagation algorithm. *Neural networks* (pp. 149-182) Springer.

Rota, P., Conci, N., Sebe, N., & Rehg, J. M. (2015). Real-life violent social interaction detection. Paper presented at the *Image Processing (ICIP), 2015 IEEE International Conference On,* pp. 3456-3460.

Shuiwang Ji, Wei Xu, Ming Yang, & Kai Yu. (2013). 3D convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 35*(1), 221-231. doi:10.1109/TPAMI.2012.59

Simonyan, K., & Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. Paper presented at the *Advances in Neural Information Processing Systems,* pp. 568-576.

Soomro, K., Zamir, A. R., & Shah, M. (2012). UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv Preprint arXiv:1212.0402,*

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. (2015). Going deeper with convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1-9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. Paper presented at the *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 2818-2826.

Wang, H., Klser, A., Schmid, C., & Liu, C. (2011). Action recognition by dense trajectories. *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference On,* pp. 3169-3176.

Weisstein, E. W. (2002). *CRC concise encyclopedia of mathematics* CRC press.

Yu, T., Kim, T., & Cipolla, R. (2010). Real-time action recognition by spatiotemporal semantic and structural forests. *Bmvc, , 2.* (5) pp. 6.