Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Detección de peleas en videos usando estimación de postura y *Bi-LSTM*

Septiembre de 2021

Tesis presentada para obtener el grado de Maestría en Ciencias de la Computación

Presenta: Jairo Egberto Powell González

Directora de tesis: Dra. Maya Carrillo Ruiz

Asesora de tesis: Dra. María de Lourdes Sandoval Solís

Índice de contenido

Índice de tablas

Índice de figuras

Resumen

Actualmente, dentro del área de visión por computadora se realizan investigaciones relacionadas con la detección de acciones y situaciones sospechosas, peligrosas o violentas. Entre ellas se encuentra la detección de peleas entre personas, en esta investigación se aborda su identificación de manera automática. Se presenta una propuesta para la clasificación de videos que contienen peleas. El método está basado en la unión de la red neuronal profunda para detección de personas *Open Pose*, y la red neuronal recurrente conocida como *Long Short Term Memory*. Se utilizaron las posturas de las personas aproximadas por *Open Pose* para calcular vectores que describen el movimiento en general de un conjunto de personas y posteriormente *Long Short Term Memory* para procesar los vectores y realizar la detección de peleas. Este método propuesto se verificó con tres conjuntos de datos etiquetados, presentes en trabajos de detección de peleas: *Movie Fight Dataset, Surveillance Camera Fight Dataset* y *Violence Detection Dataset*. La exactitud de clasificación en cada conjunto fue: 95%, 67% y 81%, respectivamente. El método propuesto se desempeñó al nivel de trabajos recientes del estado del arte y abre distintas posibilidades para realizar trabajos futuros que mejoren la exactitud y el tiempo de ejecución.

Abstract

Currently, in the area of computer vision, researchers work on the automatic detection of violent, dangerous and suspicious situations. Fights between people are among these situations, and in this work, the aim is to identify them automatically. We present a proposal of a technique for the classification of videos containing fights. The method is based in the combination of a deep neural network for human detection known as Open Pose, and the recurrent neural network known as Long Short Term Memory. Open Pose estimates postures, which we used to calculate vectors that describe the general movements of people in a video. These vectors are processed through Long Short Term Memory to classify fights. This method was verified on three labeled datasets, commonly found in fight detection research: Movie Fight Dataset, Surveillance Camera Fight Dataset and Violence Detection Dataset. Respectively, we obtained classification accuracy of: 95%, 67% and 81%. The method we proposed has a similar performance to recent works on fight detection, and it opens different possibilities for future work, to improve the accuracy of classification and execution times.

Capítulo 1 Introducción

Actualmente, dentro del área de visión por computadora se realizan investigaciones relacionadas con la detección de acciones y situaciones sospechosas, peligrosas o violentas. Entre estas situaciones se encuentran las peleas entre personas, este proyecto se enfocó en su identificación automática. Se estudiaron trabajos actuales sobre detección de peleas y violencia, realizando pruebas con algunos métodos, para proponer una contribución propia e implementarla experimentalmente.

Los algoritmos de detección de eventos de seguridad representan un apoyo al trabajo de vigilancia realizado por humanos en el presente. Existe un volumen alto de información de vigilancia que se debe procesar, como ejemplo: en la Ciudad de México existen más de 15,000 cámaras de vigilancia manejadas por el Centro de Comando, Control, Cómputo, Comunicaciones y Contacto Ciudadano [1] y ocurren hasta 200 delitos de alto impacto por día [2]. El personal humano tiene límites en el tiempo y atención que pueden dedicar a tareas de vigilancia en tiempo real, por lo cual, un algoritmo automático que se integre a su trabajo mejoraría los tiempos de respuesta en atención a incidentes. El reconocimiento automático de acciones es un área de estudio que está en constante desarrollo, y con este trabajo se busca proponer e implementar una combinación de métodos para la detección de peleas, basándose en el conocimiento y técnicas existentes, y a la vez generar una aportación a este campo.

Objetivo general

Desarrollar un método de detección de peleas en videos que sea un aporte propio al área de reconocimiento de acciones.

Objetivos específicos

a) Estudio de investigaciones de reconocimiento de acciones y detección de peleas.
b) Probar y comparar algoritmos de extracción de características usados en el área de visión de computadoras.
c) Probar y comparar algoritmos de clasificación usados en el área de visión de computadoras.
d) Diseñar y desarrollar un método que identifique la existencia de interacciones violentas entre personas.
e) Seleccionar conjuntos de datos etiquetados para entrenar y probar el método de detección.

f) Comparar los resultados obtenidos con los trabajos del estado del arte.

En este trabajo nos centramos en peleas con grupos de personas en videos de vigilancia, con conjuntos de videos etiquetados. A través de este documento se describirán las teorías, técnicas e implementaciones con las cuales se obtuvo como resultado un método de detección de peleas en videos. La técnica propuesta consiste en la detección de personas, estimación de sus posturas, abstracción del cambio general en estas posiciones y finalmente, la distinción de peleas por medio de un clasificador para información secuencial. Este método permite la representación del movimiento de un conjunto de personas con vectores. Su desempeño en cuanto a exactitud fue comparable al de trabajos actuales en el área, y puede ser adaptado, extendido y modificado para optimizar su funcionamiento, así como utilizarlo para otros trabajos en reconocimiento de acciones.

En el capítulo 2 se muestra un panorama del estado del arte en reconocimiento de acciones y detección de violencia por medio de visión de computadoras; en el capítulo 3 se describe la teoría y los algoritmos para extracción de características y clasificación, usados en esta área de investigación. Posteriormente, se detalla el funcionamiento del método de detección propuesto, en la sección 4. En el capítulo 5 se discuten los resultados de la implementación y la comparación de su funcionamiento con respecto al estado del arte. Finalmente, se presentan las conclusiones en el capítulo 6.

Capítulo 2 Estado del arte

Dentro del área de visión por computadoras, un tema de interés para la seguridad pública es la detección automática de eventos en videos; como se detalla en *"Suspicious human activity recognition: a review"* [3], algunos de estos eventos son: objetos abandonados en público, personas realizando acciones sospechosas, accidentes automovilísticos, caídas, peleas, disturbios multitudinarios e incendios.

En lo general, estos métodos funcionan en dos etapas: extracción de características y clasificación. La primera etapa consiste en transformar la información visual en datos que caracterizan los videos o imágenes y que son interpretables por computadoras, en forma de vectores y matrices. En la etapa posterior, estos datos son usados en algoritmos que automáticamente les asignan la categoría que les corresponde. En conjunto, estas dos etapas clasifican información compleja, como imágenes y videos. Diferentes trabajos usan conjuntos de ejemplos en común para evaluar y comparar sus técnicas de detección. Estos conjuntos contienen eventos que se quieren identificar positiva o negativamente, así como la etiqueta de clase. Se conocen como conjuntos de datos etiquetados.

Dentro de la detección de violencia en videos, una técnica para extracción de características es la detección de personas. Ésta se puede considerar como un problema de identificación de objetos, con dificultad añadida por la variación en la geometría y la posición de sus articulaciones. Como se menciona en Nguyen [4], las principales características que distinguen a las personas de otros objetos son la forma de los bordes, la textura, el color y el movimiento. Distintos algoritmos tienen la función de extraer estas características de imágenes y videos, por ejemplo: los Histogramas de Gradientes Orientados y *Scale-invariant Feature Transform* encuentran bordes de objetos y los asocian con ejemplos de personas en determinadas posiciones. Los *Wavelets de Haar* y Patrones Locales Binarios utilizan información de la textura de diferentes partes de la imagen para encontrar piel, ropa y cabello. En videos, se han usado Flujos Ópticos, Movimiento de Puntos de Rastreo, y Redes Neuronales Convolucionales; las cuales usan cambios en píxeles para describir movimientos en videos y detectar aquellos que correspondan a acciones determinadas.

Posteriormente a la extracción de características, se usan métodos de aprendizaje automático. Por mencionar algunos importantes: Máquinas de Soporte Vectorial, *AdaBoost*, algoritmos de agrupamiento, clasificación por Bosques Aleatorios y Redes Neuronales. Estas técnicas clasifican información representada como vectores de múltiples dimensiones sin relación temporal. Por otro lado

se encuentran los clasificadores de información secuencial, como las Redes Neuronales Recurrentes, usadas en problemas de lenguaje natural, secuencias de video, predicciones financieras, entre otros. A continuación, se describen ejemplos de trabajos recientes sobre detección de peleas, las técnicas utilizadas, el desempeño que se obtiene y sus fortalezas.

Un ejemplo de método para la detección de personas se propuso en Cao et al. [5], donde se usaron varias capas de Redes Neuronales Convolucionales combinadas para estimar posiciones de articulaciones humanas en imágenes y videos. Este sistema, conocido como *Open Pose,* se ha utilizado actualmente por su efectividad en encontrar posturas de personas y por estar disponible para uso abierto. *Open Pose* es una red profunda, por lo cual requiere recursos computacionales y tiempo considerables, pero obtiene como resultado información precisa de las posiciones de personas que puede ser almacenada como una matriz de tres dimensiones.

Seemanthini y Manjunath [6] propusieron una combinación de Segmentación de Imágenes e Histograma de Gradientes Orientados como una base para la detección de acciones y rastreo de personas en videos. La segmentación de imágenes consiste en agrupar conjuntos de píxeles que tienen colores relacionados entre sí, con el objetivo de ubicar formas de objetos. El Histograma de Gradientes es una forma de representación de las siluetas existentes en una imagen, a través de la cual se pueden identificar personas. La información de ambos métodos, en teoría se puede conjuntar para detectar personas a partir de imágenes, y posteriormente mantener un rastreo de sus movimientos durante una secuencia. Los Histogramas de Gradientes y la segmentación de imágenes tienen menor complejidad computacional que *Open Pose*, pero sus resultados son menos precisos y tienen problemas con las variaciones en rotación.

Una alternativa a la identificación de personas es la descripción de movimientos de píxeles. Como describen Naik y Gopalakrishna en *"Violence Detection in Surveillance Video-A survey"* [7] los principales métodos de esta categoría están basados en: flujos ópticos, puntos de interés, patrones binarios y matrices de co-ocurrencia de nivel de gris. Los métodos mencionados son distintas formas de observar el comportamiento de los píxeles de un video a través del tiempo, y se han usado para clasificar videos con contenido violento, como peleas y disturbios, con resultados efectivos. Estas técnicas funcionan por medio de abstracción de movimiento, a diferencia del método en el cual nos centramos, que obtiene la posición de las personas. Es más rápido de implementar y obtiene resultados exactos, pero tiene menor capacidad de generalización, a diferencia de la detección de personas.

Como se mencionó anteriormente, la evaluación y comparación de métodos se realiza por medio de conjuntos de datos etiquetados. Los más usados en detección de peleas son: *KTH, UCF-Sports, Movie Fights, UT-Interaction, Hockey Fight Database, Violent Flows* y *Violence Detection Dataset*. Cada uno de estos tiene ejemplos de peleas en contextos distintos. A continuación se describen trabajos que usan estos conjuntos de datos y técnicas basadas en detección de zonas de alto movimiento de píxeles.

En el artículo de Gao et al. [8], usando los conjuntos de datos *Hockey Fights, Violent Flows* y *Movie Fights* definieron un método llamado flujos violentos orientados, el cual encuentra las áreas de mayor movimiento de píxeles en videos. Utilizaron máquinas de soporte vectorial y AdaBoost para clasificar los datos de flujo y obtuvieron mejor desempeño que otros trabajos basados en flujo óptico, con exactitud de 88% en *Hockey Fights* y *Violent Flows Database.*

Akti et al. [9] usan diferentes Redes Neuronales Convolucionales (*CNN*) y la Red Neuronal Recurrente conocida como *Bidirectional Long-Short Term Memory (Bi-LSTM)* para experimentar con clasificación de videos de peleas en tres diferentes conjuntos de datos: *Hockey Fights*, *Movie Fights, y* un conjunto de recopilación de videos de vigilancia, obteniendo exactitud de 98%, 100% y 72%, respectivamente. Este trabajo tiene similitudes con el método que proponemos, en el uso de *Bi-LSTM,* la cual ha mostrado un desempeño óptimo para datos secuenciales. Por otro lado, su técnica utiliza Redes Neuronales que extraen información abstracta de los videos, mientras que nosotros buscamos extraer información de movimiento de personas.

Gracia et al. [10] utilizan grupos de píxeles con una cantidad alta de movimiento para realizar extracción de características más rápida que otros métodos de tipo Bolsa de Palabras. Clasifican los resultados con Máquinas de Soporte Vectorial, *AdaBoost* y Bosques Aleatorios. Supera a otros trabajos similares significativamente en el tiempo de procesamiento, con 98% de exactitud en *Movie Fights*, 82% en *Hockey Fights* y 80% en *UCF-101*. Su fortaleza fue en mejorar el tiempo de usar métodos de Bolsa de Palabras, ya que estos requieren tiempo y procesar una gran cantidad de información.

En el área de reconocimiento de acciones, Ullah et al. [11] también usaron Redes Neuronales Convolucionales y *Bi-LSTM* para clasificación en conjuntos de múltiples actividades y deportes, como *Youtube Actions, HMDB51* y *UCF-101*. Específicamente obtienen 91% de exactitud en este último.

Anteriormente, se habían utilizado descriptores de características basados en Bolsas de Palabras e Histogramas de Gradientes, por ejemplo Bermejo et al. [12], que experimentaron con estas características para evaluar su funcionamiento con los conjuntos *Movie Fights* y *Hockey Fights*.

Los métodos de las investigaciones tienen funcionamiento variable en cada conjunto de datos, pueden tener exactitud alta para un conjunto y no tenerla en otro, por sus diferencias en funcionamiento. En la tabla 1 se muestra una comparación de los trabajos anteriormente mencionados, con la exactitud de evaluación obtenida en los conjuntos que usaron; si no han usado uno, se muestra vacío.

A continuación en la Tabla 1 se resumen los métodos estudiados, el conjunto de datos utilizado y la exactitud obtenida por los mismos.

Tabla 1: Exactitud de clasificación en investigaciones de detección de peleas.

Método	*Movie Fights*	*Hockey Fights*	*UCF-101*	*Surveillance Camera Fights*	*Violent Flows*
Flujos violentos orientados (2016) [8]	-	87.5%	-	-	88%
Fight CNN y *Bi-LSTM+attention* (2019) [9]	90%	96%	-	72%	-
Xception CNN y *Bi-LSTM+attention* (2019) [9]	100%	98%	-	68%	-
Motion Blobs y Bosque Aleatorio (2015) [10]	97.8%	82.4%	79.5%	-	-
AlexNet CNN y *Bi-LSTM* (2017) [11]	-	-	91.21%	-	-
SIFT y *Histogram Intersection Kernel* (2011)*[12]*	89.5%	90.9%			

Capítulo 3 Marco teórico

En este capítulo se detallan las bases teóricas para el funcionamiento de sistemas de reconocimiento de acciones. El proceso está dividido en dos etapas: extracción de características y clasificación. Se detallarán modelos, operaciones y métodos pertenecientes al área de visión por computadoras y clasificación de datos, que son relevantes para la propuesta de este proyecto.

3.1 Extracción de características

Cuando se requiere clasificar información en forma automática, una primera etapa es la extracción de características; ésta consiste en transformar datos de un fenómeno real en representaciones que sean manejadas por computadoras, para después encontrar rasgos que identifiquen a las distintas categorías de datos.

3.1.1 Imágenes, operaciones y preprocesamiento

De acuerdo a [13], las imágenes digitales son descripciones bidimensionales discretas de imágenes reales. Una imagen digital *a[m,n]* consiste en *n* renglones y *m* columnas de píxeles, que tienen valores correspondientes a una característica visual como color o brillo. Un video, a su vez, está conformado por una secuencia de imágenes digitales, con un determinado número de cuadros por segundo.

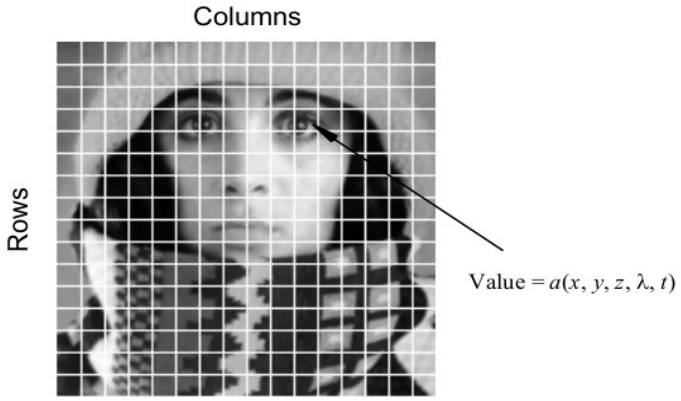


Figura 1: Imagen digital a[m,n] [13].

Dado que una imagen se puede almacenar como una matriz, sobre ella pueden aplicar operaciones aritméticas como suma, resta, multiplicación, división. Otra operación sobre imágenes relevante para este trabajo es la convolución.

La convolución es una operación en la que dos señales son procesadas para producir una nueva. En el caso de imágenes se utiliza de la siguiente forma: sea *a[m,n]* una imagen digital y *h[j,k]* una matriz que representa una ventana o *kernel*. La convolución de ambas está dada por la ecuación 1.

$$c = a \otimes h = \sum_{j=-J_0}^{J_0} \sum_{k=-K_0}^{K_0} h[j,k]\, a[m-j, n-k] \qquad (1)$$

donde *[j=0, k=0]* es el centro de la matriz *h*, y J_0 y K_0 representa la distancia de los límites de la matriz al centro. Cuando se aplica un filtro o *kernel* específico a una imagen se pueden obtener diferentes efectos, por ejemplo: suavizar una imagen, realizar una operación matemática o una derivada, o filtrar frecuencias. En este trabajo se usaron dos filtros en los experimentos: Gaussiano y Sobel; que respectivamente suavizan y obtienen la primera derivada de la imagen. El efecto de estos filtros se puede observar en la figura 2.

Figura 2: Ejemplos de filtros: Imagen normal, Filtro Gaussiano y Filtro Sobel. [13]

Este tipo de operaciones se utilizan para mejorar imágenes, transformarlas y extraer información importante de ellas.

3.1.2 Redes convolucionales

Tripathi et al. [14] describen a las redes neuronales como estructuras basadas en el funcionamiento del cerebro, que funcionan como una extensión de la regresión logística. La unidad básica de estas redes es la neurona, que tiene la función de recibir un vector de datos *x* y asignarle una categoría binaria por medio de operaciones internas. Cuando se usa un conjunto de neuronas se obtiene una estructura capaz de clasificar información más compleja. Las redes neuronales tienen múltiples formas de construirse con diferentes arquitecturas y operaciones para diferentes aplicaciones y problemas. En la figura 3 se observa la funcionalidad básica de una neurona.

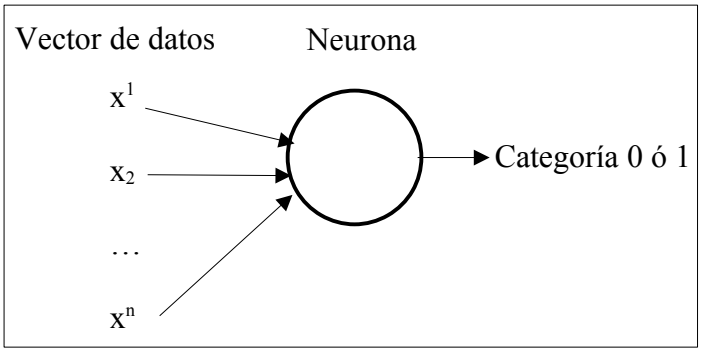


Figura 3: Neurona artificial.

Cuando se integran operaciones de convolución con la funcionalidad de una red neuronal, se obtienen las redes neuronales convolucionales. Éstas procesan una imagen a través de distintas capas de convolución para obtener un vector de las características relevantes para el problema. Posteriormente se usan redes neuronales para clasificar u obtener determinados datos sobre la imagen.

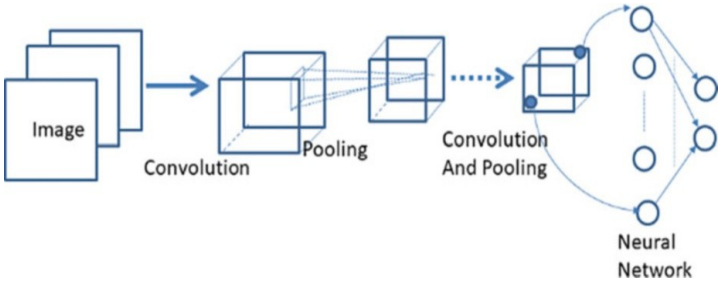


Figura 4: Red convolucional [14].

3.1.3 Open Pose

Open Pose es una red neuronal profunda, creada por Cao *et al.* [5], que detecta personas en imágenes y videos; está conformada por múltiples capas de redes neuronales convolucionales que procesan una imagen para obtener la postura corporal de la gente en ella.

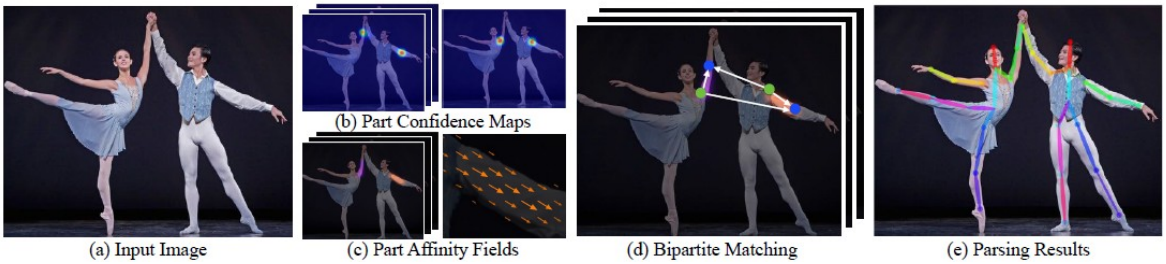


Figura 5: Funcionamiento general de Open Pose [5]

Como se observa en la figura 5, la estructura de *Open Pose* está construida a partir de dos redes convolucionales: una de ellas identifica puntos de articulación del cuerpo, mientras que la otra encuentra las conexiones de articulaciones. Ambas redes parten de una imagen y calculan matrices que se conocen como "mapas de calor", cuya función es indicar zonas con alta probabilidad de ser articulaciones o miembros. Posteriormente estos mapas de calor son procesados para obtener las posiciones de cada persona.

Cada mapa de calor es una matriz cuyos valores representan la probabilidad de que un área correspondiente en la imagen sea una articulación o miembro específico. Los mapas son divididos en dos tipos: "Mapas de Confianza de Partes" y "Campos de Afinidad de Partes". Los Mapas de Confianza representan las articulaciones, mientras que los Campos de Afinidad representan los miembros que las conectan.

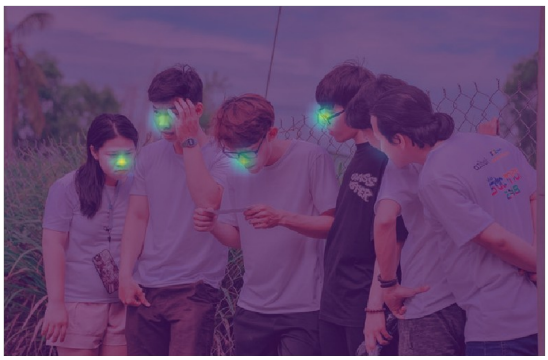


Figura 6: Mapa de Confianza para narices. [14]

Figura 7: Campos de Afinidad para Brazo [14]

Las partes del cuerpo obtenidas dependen del modelo utilizado, y se obtiene un Mapa de Confianza o un Campo de Afinidad para cada una; como se observa en las figuras 6 y 7, donde las partes que se buscan están resaltadas gráficamente según su valor de probabilidad en el mapa de calor. Se pueden tener dos modelos de cuerpo completo: COCO y MPI. Las partes del cuerpo en cada modelo son las siguientes:

Tabla 2: Partes del cuerpo y números de identificación

Identificador	Modelo MPI	Modelo COCO
0	Cabeza	Nariz
1	Cuello	Cuello
2	Hombro derecho	Hombro derecho
3	Codo derecho	Codo derecho
4	Muñeca derecha	Muñeca derecha
5	Hombro izquierdo	Hombro izquierdo
6	Codo izquierdo	Codo izquierdo
7	Muñeca izquierda	Muñeca izquierda
8	Cadera derecha	Cadera derecha
9	Rodilla derecha	Rodilla derecha
10	Tobillo derecho	Tobillo derecho
11	Cadera izquierda	Cadera izquierda
12	Rodilla izquierda	Rodilla izquierda
13	Tobillo izquierdo	Tobillo izquierdo
14	Pecho	Ojo derecho
15	Fondo	Ojo izquierdo
16		Oreja izquierda
17		Oreja derecha
18		Fondo

El proceso de obtener las posiciones de las personas a partir de los mapas de calor es descrito por Gupta [14]. Primero se transforman los Mapas de Confianza en imágenes binarias, donde las zonas con mayores valores forman un grupo conocido como *blob*. En la figura 8 se muestra un ejemplo de imagen binaria.

Figura 8: Blobs de mapas de confianza [14]

En cada *blob* se realiza una búsqueda del valor máximo y sus coordenadas se consideran el punto central de la articulación que represente.

Los Campos de Afinidad definen conexiones entre articulaciones, por lo cual se aplica un proceso similar con diferente objetivo. En lugar de un punto, se determina una línea recta que conecte dos articulaciones.

Los puntos y las líneas sirven para construir los esqueletos de la posición de las personas. Las articulaciones se unen entre sí por pares, por medio de la línea que les corresponda. Las partes del cuerpo serán asignadas a una persona según se hagan las conexiones. En la figura 9 se observa el proceso concluido, donde cada persona está representada por las coordenadas y conexiones que le pertenecen. Esta información puede ser almacenada en matrices para realizar clasificación posteriormente.

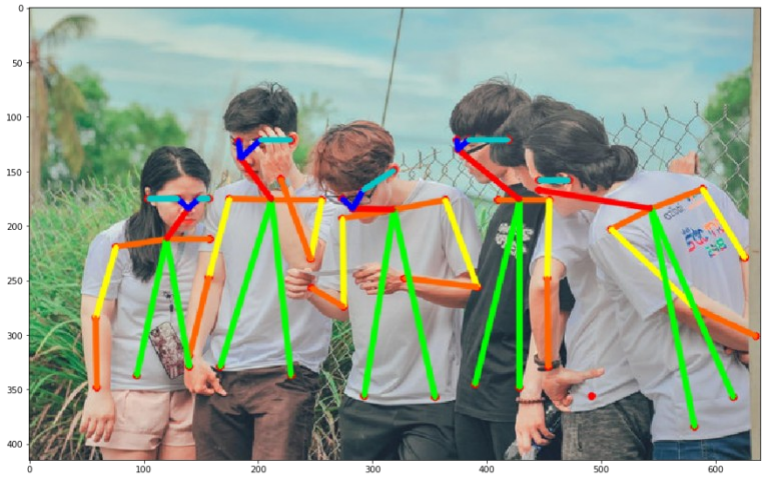


Figura 9: Resultados de Open Pose. [14]

3.2 Clasificación

Una vez que se han extraído características se utilizan algoritmos para encontrar automáticamente patrones en los datos y asignarles una clasificación. Existen múltiples métodos de clasificación con diferentes características, complejidad, ventajas y desventajas. Éstos se basan en conjuntos de datos etiquetados, que consisten en vectores de características *x* pertenecientes a una categoría conocida *y.* En distintas maneras, se aprende a clasificar nuevos datos desconocidos por medio de los etiquetados.
Los algoritmos usados fueron: Máquinas de Soporte Vectorial, *AdaBoost*, Bosques Aleatorios y *Long Short Term Memory* bidireccional (*Bi-LSTM)*. Los primeros tres métodos se usan para comparar con el último, que es el clasificador seleccionado.

3.2.1 Máquinas de Soporte Vectorial

Como se explica en *Introduction to Machine Learning*[14], las Máquinas de Soporte Vectorial (SVM) son clasificadores lineales binarios. Suponiendo que se tiene un conjunto de datos etiquetados en forma de vectores multidimensionales, y se pueden dividir en dos categorías de manera lineal, las *SVM* encuentran una separación óptima.
SVM tiene el siguiente modelo matemático (Ecuación 2): Sea *x* un vector de características, *w* un conjunto de valores de peso y *b* una constante conocida como *bias* o sesgo, se define una línea:

$$wx + b \tag{2}$$

Donde las categorías corresponden a las dos partes del espacio divididas por el hiperplano. Pueden existir múltiples líneas diferentes que separen dos clases y el objetivo de *SVM* es encontrar aquella que maximice el margen de separación.
Las *SVM* se han adaptado para clasificar más de una categoría, así como algunos problemas no lineales. Son “una forma de clasificación eficiente y efectiva con cantidades pequeñas de datos” [14, p. 66].

3.2.2 AdaBoost

AdaBoost es un algoritmo diseñado por Freund y Schapire [16], cuyo funcionamiento se basa en utilizar algoritmos de aprendizaje "débiles" de forma estratégica, para construir un algoritmo de aprendizaje mejorado.

Un conjunto de datos etiquetados se divide en diferentes distribuciones. Cada distribución se aprende a clasificar por medio de un algoritmo débil. Éste calcula una hipótesis *h(x)* para un dato *x,* que tendrá un error respecto a la categoría verdadera *y*. Se combinan las hipótesis de los algoritmos débiles para formar un mejor clasificador.

De acuerdo a Domingo y Watanabe [17], aunque *AdaBoost* ha sido una de las técnicas de mejora más exitosas que se han diseñado, sus desventajas principales son la susceptibilidad al ruido y errores durante el proceso que podrían evitar una generalización correcta, es decir, predecir correctamente las categorías de datos desconocidos.

3.2.3 Random Forests

Los *Random Forests* o bosques aleatorios son colecciones de árboles de decisión que realizan predicciones basadas en datos etiquetados, como describen Gopinath *et al.* [14]. Un árbol de decisión es una estructura que representa un proceso de predicción basado en nodos y aristas. Un nodo es una variable, que usualmente tiene dos aristas que representan posibles decisiones. Al iniciar con una variable y conformar un conjunto de ramificaciones se construye un árbol, que asigna una categoría basado en las decisiones tomadas.

Individualmente este método de clasificación puede tener problemas para obtener resultados óptimos, por lo cual se crearon los bosques aleatorios; éstos dividen los conjuntos de datos en muestras que son clasificadas por diferentes árboles de decisión y se hace un agregado de las predicciones para asignar una clasificación final. El modelo de predicción funciona de la siguiente forma: Sea x un dato y f_n el resultado obtenido por un árbol de decisión n, el resultado final del bosque aleatorio será el promedio de los árboles, como se observa en la ecuación 3.

$$f = \frac{1}{B} \sum_{n=1}^{B} f_n(x) \qquad (3)$$

Los árboles de decisión son modelos “efectivos e intuitivos para problemas de clasificación y regresión” [14 p. 77]. Sin embargo, sus principales desventajas son: Cuando las variables tienen un alto número de posibles valores, el modelo obtenido es menos efectivo debido al número de combinaciones que se forman; los datos nuevos no pueden aprenderse en forma incremental, sino que se debe volver a entrenar el bosque; y son más lentos que otros clasificadores, particularmente con mayor tamaño de variables y árboles.

3.2.4 Redes neuronales Recurrentes

Las redes neuronales recurrentes (*RNN*) son un tipo de red neuronal específicamente diseñadas para manejar datos en secuencia, lo cual es útil en aplicaciones como procesamiento de lenguaje, predicciones financieras y clasificación en videos. Como describe Olah [18], las *RNN* procesan una secuencia de datos x_i. En un momento i, una neurona recibe la información actual x_i y la información pasada con el resultado de la neurona anterior. Al final de la secuencia, se asigna una clasificación. En la figura 10 se puede observar la estructura general de una red neuronal recurrente.

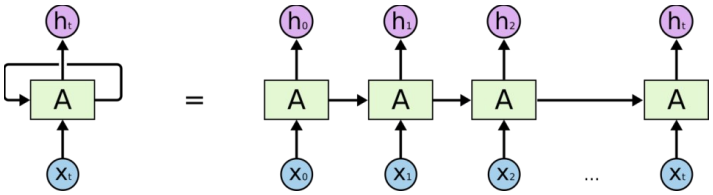


Figura 10: Red neuronal recurrente. [18]

Las *RNN* tienen una limitación dado que no manejan “dependencias de largo plazo”, es decir, información pasada, que es importante para el presente, pudo haberse perdido al procesarse durante la secuencia. Hochreiter y Schmidhuber [19] crearon las redes *Long Short Term Memory (LSTM)* para solucionar el problema de almacenamiento de información.

Las redes *LSTM* son una modificación de las *RNN* que seleccionan información para mantener en una “memoria”. Las neuronas de las redes recurrentes comunes contienen una capa que procesa la entrada y la salida del tiempo anterior; mientras que las de *LSTM* tienen más funciones dentro de la neurona: Seleccionar, filtrar y almacenar datos. La diferencia entre los dos tipos de red se observa en la figura 9.

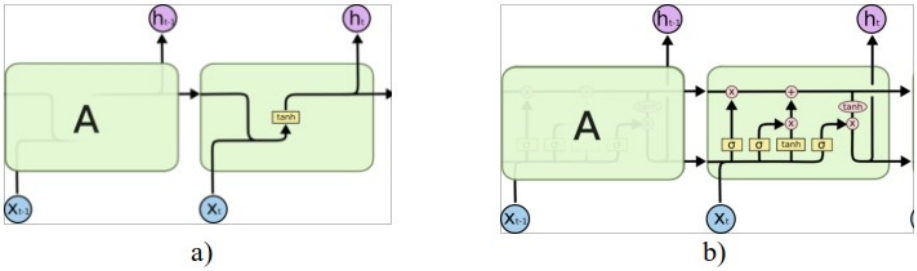


Figura 11: a) Red Neuronal Recurrente. b) Long Short Term Memory. [18]

Suponiendo que se tiene un conjunto de vectores de características x_i *(i=0...t)* organizados en el tiempo, y que *t* es un tamaño de secuencia fijo, cada vector se procesa en la neurona correspondiente a su tiempo *i*. La entrada a la neurona se procesa en tres capas con diferentes funciones, que se explicarán a continuación.

LSTM tiene un módulo denominado estado de célula *(Cell state)* y a través de éste se mantiene información selecta a lo largo de todas las neuronas. El estado de célula se puede observar en la figura 12.

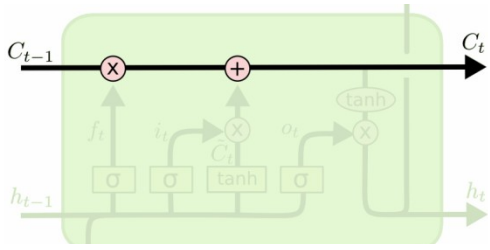


Figura 12: Estado de célula. [18]

La primera capa decide si se almacenan valores en el estado de célula. Esta capa funciona por medio de una función matemática que entrega como salida un valor entre 0 y 1, después de procesar el vector x_i y la salida de la neurona anterior h_{i-1}; donde 1 equivale a mantener toda la información de entrada y 0 a desecharla.

En la segunda capa se tienen dos funciones: decidir cuanta información mantener en el estado de célula y crear un vector de valores candidatos para añadirle. La información que se decidió recordar en la primera capa se multiplica por los valores obtenidos en la segunda capa; esto formará el estado de célula actualizado.

La última capa procesa el vector de tiempo actual x_i, la salida de la neurona anterior h_{i-1}, y la información C_t contenida en el estado de célula. Se obtiene la salida h_i para la neurona siguiente o como categoría final h_t.

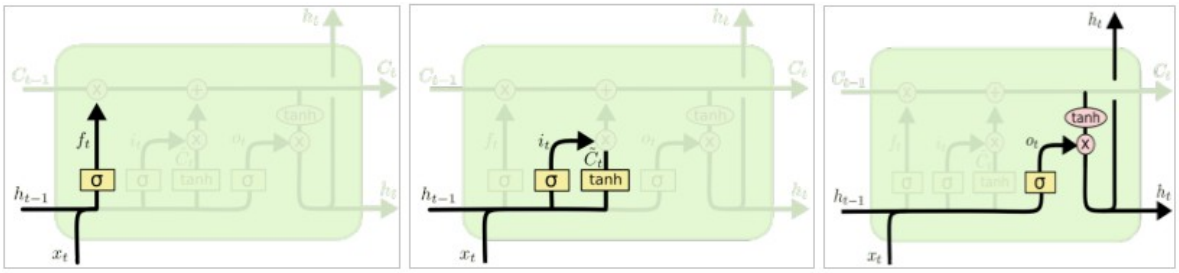


Figura 13: Primera, segunda y tercera capa de neurona. [18]

3.2.5 LSTM Bidireccional

Una red *Bi-LSTM* es una estructura formada por dos capas de redes *LSTM*, en las cuales se procesa la misma secuencia en orden distinto para mejorar el desempeño en clasificación. Una red recibe la información en orden normal: x_i, *(i=0, 1, 2...t),* mientras que la segunda red invierte el orden de los datos: x_i, *(i=t, t-1, t-2, ... ,0)*. Después los resultados de ambas redes se integran, obteniendo una clasificación de mayor exactitud que con una red singular [9] [11].

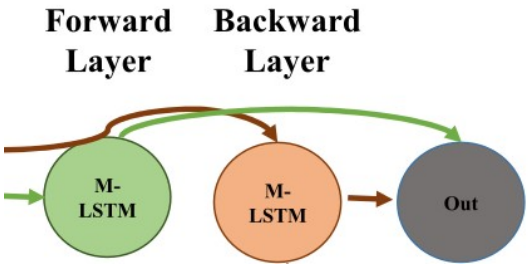


Figura 14: Ejemplo de red LSTM Bidireccional [11].

Capítulo 4 Metodología propuesta

En esta sección se describirá cada una de las partes de la técnica propuesta para detección de peleas, desde el procesamiento de los videos hasta el clasificador. Primero se explicará el funcionamiento general del método y posteriormente se detallará cada paso en el proceso de clasificación.

4.1 Método de detección

El proceso de clasificación tiene dos etapas principales: detección de posturas por medio de *Open Pose* y clasificación por *Bi-LSTM*. Se requirieron distintos pasos intermedios para ajustar la información, y es la integración de éstos con los algoritmos donde se encuentra el aporte de este proyecto. El funcionamiento general del método es el siguiente:

1. Extraer un número de cuadros de los videos.
2. Realizar preprocesamiento a cada imagen.
3. Procesar los cuadros por *Open Pose* para detectar posturas.
4. Transformar las matrices de posturas en vectores de frecuencia de ángulos.
5. Entrenar una red *Bi-LSTM* con las secuencias de vectores.
6. Evaluar la exactitud de clasificación de los tres conjuntos de datos.

En la figura 15 se esquematiza esta propuesta.

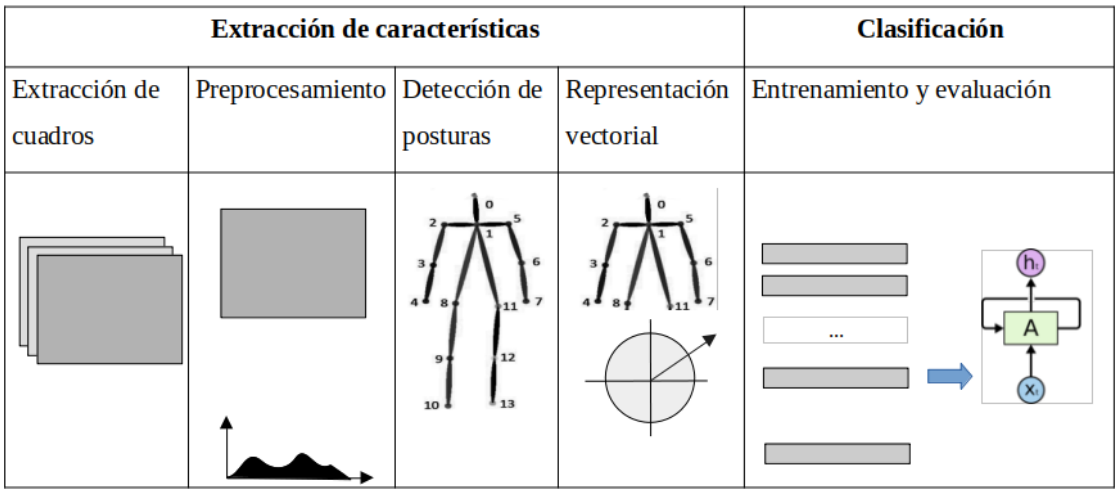


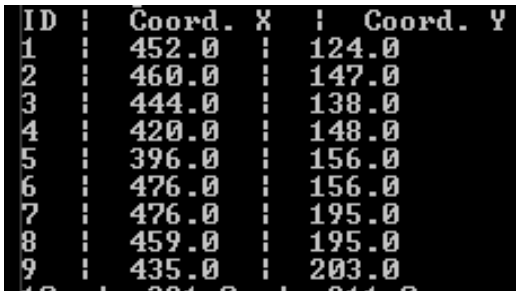
Figura 15: Método propuesto

4.1.1 Extracción de cuadros

Se utilizó *OpenCV* para realizar operaciones sobre los videos, la cual es una de las librerías de código abierto más usadas en visión de computadoras. Considerando que los videos de los conjuntos de datos tienen longitud de entre 2 y 4 segundos, se eligió extraer 10 cuadros igualmente espaciados en el tiempo; de este modo se toma un cuadro cada fracción de segundo, suficiente para representar adecuadamente las características principales dentro de la secuencia completa. Para mantener la información de manera uniforme, se ajustaron las imágenes extraídas a una resolución de 640 píxeles de alto y 360 de ancho; ésta mantiene la calidad de la imagen para su procesamiento por *Open Pose,* y requiere un tiempo de procesamiento menor que el necesario para mayores resoluciones. Las imágenes se ordenaron por *dataset*, categoría y número de video para después mejorarse con filtros y estimar las posturas de personas por medio de *Open Pose.*

4.1.2 Extracción de posturas

Como se mencionó en el capítulo anterior, *Open Pose* es una red profunda que calcula mapas de calor de partes del cuerpo para la estimación de posición, y se utilizó la implementación de código en *Python* descrita por Gupta [14] para recuperar las posturas de las personas y representarlas en matrices. Esta red tiene complejidad computacional relativamente alta, pero la representación obtenida usa poco espacio en memoria y es precisa para el objetivo de extraer movimientos de partes del cuerpo. Se usó el modelo COCO, y el procedimiento tiene como resultado dos matrices. La primera matriz contiene un índice de partes del cuerpo con su número de identificación, coordenadas y el valor de confianza. La segunda matriz contiene el número de personas y el identificador que relaciona cada parte del cuerpo con la persona correspondiente.



ID	Coord. X	Coord. Y
1	452.0	124.0
2	460.0	147.0
3	444.0	138.0
4	420.0	148.0
5	396.0	156.0
6	476.0	156.0
7	476.0	195.0
8	459.0	195.0
9	435.0	203.0

Figura 16: Ejemplo de matriz de coordenadas para una persona.

Entonces, para cada imagen quedaron representados los esqueletos de todas las personas en una matriz de tamaño *[n, 18, 2]*, donde *n* es el número de personas detectadas, con las coordenadas de sus 18 miembros respectivos dentro de la imagen. En la figura 16 se muestra un ejemplo de la información en la matriz. El primer número es el identificador de parte, mientras que los dos valores siguientes son las coordenadas en los ejes *X* y *Y*.

4.1.3 Preprocesamiento de imágenes

Muchos de los videos en los conjuntos de datos tienen condiciones visuales inadecuadas para *Open Pose,* por las diferencias en resolución y contexto en que se filmaron, entonces se realizaron experimentos de preprocesamiento para mejorar los resultados de detección de personas. Se utilizaron diferentes combinaciones de operaciones como: modificación de brillo y contraste, adición y sustracción de imágenes, suavizado, y detección de bordes. Por medio de la librería *OpenCV* se aplicaron estas operaciones sobre los cuadros de videos con diferentes configuraciones. Se obtuvo una mejora significativa utilizando una combinación de detección de bordes con adición de imágenes: Se resaltaron las siluetas de los objetos en la imagen y se combinaron con la foto original, con lo cual se mejoró la distinción de objetos y se resolvieron problemas como bajo contraste y falta de nitidez.

La detección de bordes se realizó por convolución, la cual asigna valores altos de brillo a zonas de la imagen donde hay cambios de color; de modo que puede usarse para observar las siluetas existentes en ella. Se probaron diferentes filtros y se eligió uno conocido como Sobel, dado que sus resultados mejoraban la imagen sin distorsionarla. En la ecuación 4 se muestra un filtro de Sobel que realiza una operación de diferencias de derecha a izquierda.

$$h_{sobel} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad (4)$$

Cuando a una imagen se aplica convolución con este filtro, se obtiene una imagen en blanco y negro con siluetas marcadas. Este filtro es direccional, es decir que detecta los bordes recorriendo la imagen en una dirección, según la forma de la matriz de convolución: derecha a izquierda, arriba a abajo, o sus inversos. Se observó información más completa al realizar un filtrado en una dirección, otro en la dirección opuesta y combinar las imágenes de bordes en una misma.

Dos imágenes pueden añadirse en forma de una combinación lineal. Los valores en un píxel de una imagen *a* se suman en forma ponderada con el píxel correspondiente en una imagen *b*, como se muestra en la ecuación 5.

$$c(x,y) = k \cdot a(x,y) + (1-k)\, b(x,y) \quad (5)$$

Entonces en el método propuesto, una imagen original pasa por el siguiente proceso, el cual se determinó en forma experimental: Se crean dos imágenes de bordes por medio de filtros horizontales de Sobel en direcciones opuestas. Ambas imágenes se añaden en la misma proporción, es decir *k=5,* formando una imagen más completa de los bordes. Después ésta última se añade a la imagen original, en proporción 30/70, *k =0.7* para la imagen original y *1-k=0.3* para la imagen de bordes. El efecto de mejora en la detección se observa en las siguientes figuras.

Figura 17: Open Pose ejecutado en imágenes sin preprocesamiento. [25]

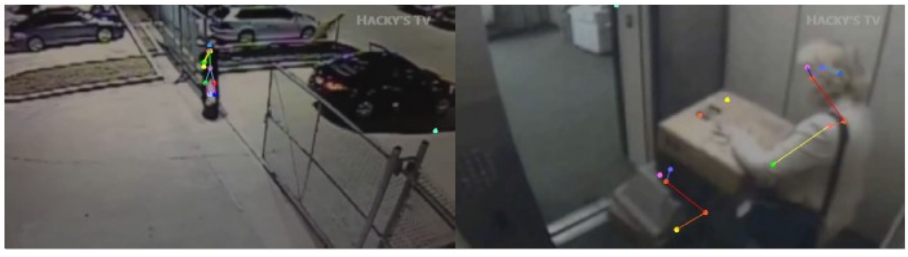


Figura 18: Open Pose ejecutado en imágenes con el preprocesamiento propuesto. [25]

4.1.5 Representación de personas

Como se describió anteriormente, la ejecución de *Open Pose* sobre una imagen da como resultado una matriz de tamaño *[n, 18, 2]*. Los videos de los conjuntos de datos tienen un número variable de personas, incluso dentro de un mismo video, por lo cual el tamaño de la matriz es también variable. Sin embargo, los algoritmos de clasificación requieren vectores de tamaño consistente para funcionar. Por lo tanto, fue necesario buscar una manera de transformar la información extraída de posturas en un vector de datos con tamaño igual, independientemente del número de personas en cada cuadro. Existirían dos maneras de manejar los datos: aplicar un algoritmo de clasificación a cada matriz de persona en los videos, o comprimir la información de todas las personas en un vector de características. La primera opción tiene mayor complejidad de implementación, porque requiere rastrear a cada persona a lo largo de la secuencia, clasificar el cambio en sus posturas y, adicionalmente, usar la información conjunta de las personas para categorizar el video como violento o no violento. La segunda opción tiene menor complejidad que la primera, pero requeriría un método que extraiga adecuadamente las principales características en las posturas de las *n* personas. El funcionamiento teórico de ambos métodos se muestra en las figuras 19 y 20, se eligió la segunda opción.

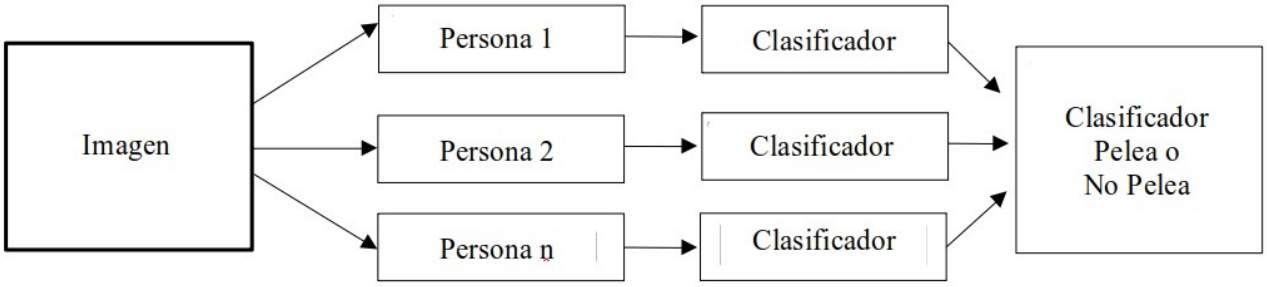


Figura 19: Clasificación individual de acciones de personas.

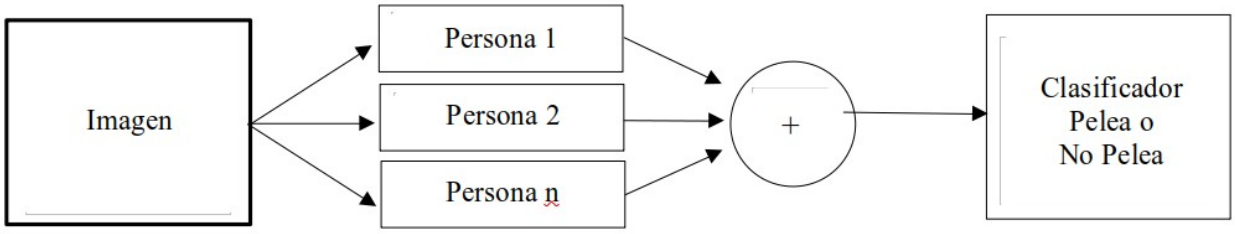


Figura 20: Clasificación en conjunto de acciones de personas

La información de las posturas de las personas está expresada en las coordenadas de sus articulaciones y las conexiones entre ellas. Un dato que se puede extraer directamente es el ángulo que forman sus miembros. Esto podía ser un atributo que representara información de las personas para clasificar acciones violentas.

Para identificar una pelea, los miembros más relevantes serían los brazos, las piernas y el torso, ya que están mayormente involucradas en acciones agresivas. Si la posición de una persona está representada por un conjunto de puntos y aristas que conectan pares, entonces se puede calcular para cada arista el ángulo que forma respecto al eje horizontal. Las partes del cuerpo relevantes están representadas de la manera que se muestra en la figura 21, con 14 puntos y 13 aristas extraídas del modelo de 18 puntos obtenido con *Open Pose.*

Arista	Articulaciones
(0, 1)	Nariz a cuello
(1, 2)	Cuello a hombro derecho
(2, 3)	Hombro derecho a codo derecho
(3, 4)	Codo derecho a muñeca derecha
(1, 5)	Cuello a hombro izquierdo
(5, 6)	Hombro izquierdo a codo izquierdo
(6, 7)	Codo izquierdo a muñeca izquierda
(1, 11)	Cuello a cadera izquierda
(11, 12)	Cadera izquierda a rodilla izquierda
(12, 13)	Rodilla izquierda a tobillo izquierdo
(1, 8)	Cuello a cadera derecha
(8, 9)	Cadera derecha a rodilla derecha
(9, 10)	Rodilla derecha a tobillo derecho

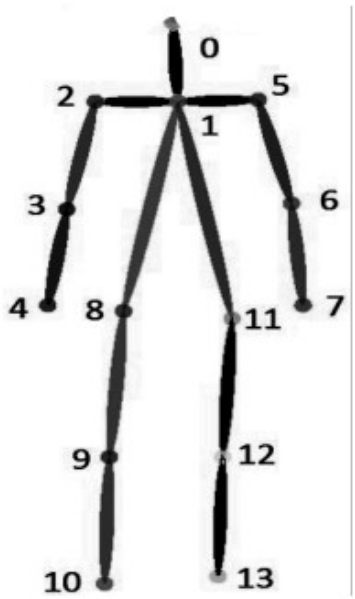


Figura 21: Representación de una persona.

El ángulo de cada arista se calculó en forma aproximada para facilitar el almacenamiento. En lugar de utilizar funciones trigonométricas, se dividió un círculo en 20 ángulos igualmente espaciados y se

asoció la arista al ángulo más cercano. Así, las características de la arista pueden almacenarse en vectores fijos que representan ángulos predeterminados y no es necesario almacenar un ángulo específico para cada arista.

Para cada arista que representa una parte del cuerpo, se calcula el ángulo predeterminado más cercano y se añade como un conteo en un mismo vector para las *n* personas del cuadro. De modo que en un vector de tamaño 20 se tiene la frecuencia de orientaciones de una extremidad para todos los sujetos en una imagen. Su importancia yace en que es una forma de abstraer el comportamiento general de un miembro por medio de su ángulo, por ejemplo: En un video donde se ven personas caminando normalmente, los brazos y piernas estarán en dirección del suelo, haciendo movimientos periódicos; mientras que, en una pelea, cambiarán drásticamente y tendrán un comportamiento diferente que podría ser identificado por medio de sus orientaciones. Se eligieron 20 intervalos de ángulos para tener un tamaño de vector que representara suficientes variaciones de movimiento. Se podría aumentar su exactitud, a cambio de mayor complejidad de los vectores.

Entonces se aplica el mismo proceso para las 13 aristas seleccionadas, obteniendo un vector de 13 miembros, con 20 ángulos predeterminados cada uno. Esto forma un vector de tamaño 260 con conteos de frecuencia de ángulos. Posteriormente estos conteos de frecuencia se escalan linealmente para tener valores de distribución entre 0 y 1.

En resumen, para representar un video:

1. Se extrajeron 10 cuadros.
2. Los cuadros se procesan por medio de filtros Sobel y adición de imágenes.
3. A cada cuadro se extraen características con *Open Pose*, obteniendo una matriz de *n* personas *[n, 18, 2]*.
4. Las matrices de personas se transforman en vectores de tamaño 260 que contienen una distribución de frecuencia de ángulos.

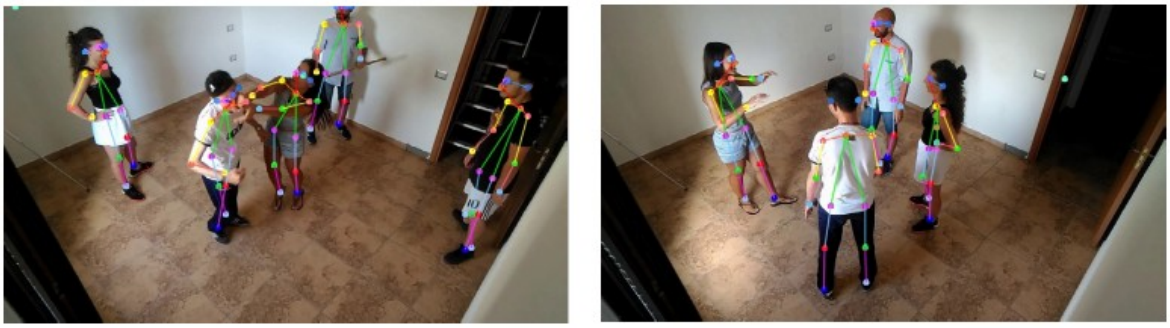


Figura 22: Ejemplo de extracción de posturas en videos con peleas y sin peleas. [26]

4.1.6 Clasificación

Para clasificar la información obtenida se usó la red neuronal *Bi-LSTM*, la cual se desempeña de mejor manera con información secuencial, por ejemplo: videos, lenguaje natural y datos financieros. Como se mostrará posteriormente se obtienen mejores resultados que con el uso de clasificadores de vectores sin relación temporal. Para la implementación de la red *Bi-LSTM* se utilizó la librería *Tensorflow* para *Python*, que maneja arquitecturas de redes neuronales de forma flexible. Se implementó directamente una red, con los siguientes parámetros básicos: número de neuronas o "unidades ocultas", tamaño de vectores de entrada, tamaño de secuencia, épocas de entrenamiento y número de categorías. El tamaño de vector descrito es de 260 datos, la secuencia tiene 10 cuadros distintos y hay dos categorías: Pelea y No-Pelea. El número de épocas de entrenamiento y de neuronas puede variar según el conjunto de datos que se utilice y el comportamiento que se quiera dar a la red. Observamos un comportamiento estable con 520 neuronas, y con más de 10 pero menos de 20 épocas de entrenamiento, dependiendo del conjunto de datos; esto evita que la red se sobre-entrene.

Se entrenó y probó la red *Bi-LSTM* con los datos extraídos de los conjuntos de videos, por separado. Cada conjunto se dividió proporcionalmente en 80% de los videos para entrenamiento y 20% para evaluación. La división se realizó aleatoriamente con varias repeticiones, para observar el comportamiento promedio.

En el siguiente capítulo se encuentran los detalles de la implementación, los experimentos que fueron realizados y los resultados que se obtuvieron.

Capítulo 5 Implementación

En este capítulo se describen las características de los conjuntos de videos, los experimentos realizados y los resultados que se obtuvieron en cada uno de los conjuntos. Se discuten los resultados, comparándolos con trabajos que usaron los mismos conjuntos de datos y analizando el funcionamiento de este método en contraste con los otros.

El método propuesto en este proyecto utiliza *Open Pose* como algoritmo de extracción de características de videos y una red *Bi-LSTM* para clasificar. Ambos métodos se implementaron por medio de las siguientes librerías en el lenguaje *Python 3.8*. Todo el software utilizado es de código abierto para la posible reproducción de los experimentos.

- *Open Pose* 1.6.0 (*CPU Release*) [20]: Librería para implementar la red *OpenPose*.
- *Open CV* 4.5.1.48 [21]: Librería de visión de computadoras.
- *Tensorflow* 2.4 [22]: Librería para aprendizaje automático.
- *Scikit-Learn* 0.24.0 [23]: Librería de aprendizaje automático.

Los conjuntos de datos seleccionados para entrenar los clasificadores fueron tres, *Movie Fights Dataset, Surveillance Camera Fight Dataset* y *Violence Detection Dataset.*

5.1 Movie Fight Detection Dataset

Este conjunto de datos creado por Nievas *et al.* [24] en 2011 consiste en 200 videos etiquetados extraídos de películas. La mitad de los videos están etiquetados como peleas y la otra mitad como no-peleas. Todos los videos tienen duración de dos segundos, con variaciones en la resolución y el número de cuadros por segundo. Las peleas están coreografiadas, con ángulos determinados por la filmación y calidad dependiente de la película de la cual se extrajo.

Figura 23: Ejemplo de video. [24]

5.2 Surveillance Camera Fight Dataset

El conjunto creado por Akti *et al.*. [25] en 2019 está conformado por 150 videos de peleas y 150 sin peleas. Las secuencias fueron extraídas de situaciones reales captadas por cámaras de vigilancia y contienen peleas con un número variable de personas, lugares y ángulos. Los videos tienen duración variable y menor a 5 segundos, así como diferentes resoluciones y tasa de cuadros por segundo. La calidad de las imágenes en este conjunto motivó a la experimentación con filtros de mejora.

Figura 24: Videos del conjunto de datos [25].

5.3 Violence Detection Dataset

Este conjunto fue creado en 2020 por Bianculli *et al.* [26]. para probar sistemas de detección de peleas. Tiene 230 videos con peleas y 120 sin peleas, grabados con cámaras fijas. De acuerdo con los autores, un número variable de actores fueron grabados simulando diferentes acciones violentas; y para los videos no violentos, realizando acciones que pudieran resultar en falsos positivos por la velocidad o la similitud con acciones agresivas. Los videos tienen una resolución de 1920 x1080 píxeles, tasa de 30 cuadros por segundo y longitud promedio de 5 segundos.

En la siguiente tabla se muestra el número de videos que tiene cada conjunto de datos, según su categoría.

Tabla 3: Número de videos en conjuntos de datos.

Dataset	Secuencias con peleas	Secuencias sin peleas
Surveillance Camera Fight Dataset	150	150
Violence Detection Dataset	230	120
Movie Fights Dataset	100	100

5.4 Experimentos y Resultados

Los tres conjuntos de video se entrenaron y evaluaron por separado. Un parámetro del experimento fue realizar clasificación con y sin preprocesamiento de imágenes. Para observar el funcionamiento del método propuesto con otros métodos de clasificación, además de *Bi-LSTM* se usaron Máquinas de Soporte Vectorial (*SVM*), Bosques Aleatorios (*RF*) y *AdaBoost*. Usando el 80% de los datos de cada conjunto, se entrenaron los métodos de clasificación hasta obtener una clasificación correcta. Posteriormente, el 20% de las secuencias se usaron para evaluación de la capacidad de predicción. Para ello, se predijeron su categoría con los clasificadores entrenados y se calculó la exactitud comparando la categoría predicha con la etiqueta correcta. En la ecuación 5 se muestra como es calculada la exactitud, dados el número de verdaderos positivos (VP), verdaderos negativos (VN), falsos positivos (FP) y falsos negativos (FN). Estos positivos y negativos corresponden a las categorías del problema: Pela y No-Pelea.

$$Exactitud = \frac{VP + VN}{VP + VN + FP + FN} \tag{5}$$

Se realizó validación cruzada a diez pliegues con los mismos porcentajes de entrenamiento y evaluación, con el conjunto de secuencias siendo dividido en forma aleatoria cada repetición. Los

resultados de exactitud se promedian y en las siguientes tablas se muestran los resultados obtenidos para cada experimento y conjunto de datos.

Tabla 4: Resultados con Surveillance Camera Fight Dataset.

Dataset	Preprocesamiento	Clasificador	Exactitud
Surveillance Camera Fight Dataset	Sin filtros	*SVM*	46%
		RF	48%
		AdaBoost	56%
		Bi-LSTM	51%
	Con filtros	*SVM*	65%
		RF	61%
		AdaBoost	60%
		Bi-LSTM	67%

Tabla 5: Resultados con Violence Detection Dataset.

Dataset	Preprocesamiento	Clasificador	Exactitud
Violence Detection Dataset	Sin filtros	*SVM*	68%
		RF	69%
		AdaBoost	67%
		Bi-LSTM	81%
	Con filtros	*SVM*	69%
		RF	70%
		AdaBoost	71%
		Bi-LSTM	81%

Tabla 6: Resultados con Movie Fights Dataset.

Dataset	Preprocesamiento	Clasificador	Exactitud
Movie Fights Dataset	Sin filtros	*SVM*	90%
		RF	89%
		AdaBoost	84%
		Bi-LSTM	95%
	Con filtros	*SVM*	91%
		RF	89%
		AdaBoost	92%
		Bi-LSTM	95%

En los conjuntos de videos algunas de las secuencias causaron dificultad para que *Open Pose* realizará su proceso y tras experimentación con filtros se mejoró la detección con el preprocesamiento descrito en el capítulo anterior. Como se observa en las Tablas 4 a 6, el uso de filtros mejora la detección de personas, y con ello la exactitud con los 4 algoritmos de clasificación.

En todos los experimentos, la red *Bi-LSTM* se desempeñó mejor que los otros tres métodos, debido a que está diseñada para manejar datos secuenciales, a su estructura bidireccional y su capacidad de memoria.

El *Surveillance Camera Fight Dataset*, creado y probado por Akti *et al.* [9] tiene el método siguiente: Una red neuronal convolucional conocida como Fight, diseñada específicamente por los autores para detectar movimientos correspondientes a peleas; y una red *Bi-LSTM* con funcionalidades añadidas llamadas “atención”. Usando diferentes redes convolucionales y configuraciones de parámetros, su exactitud varía hasta 10 puntos en los diferentes conjuntos con los que prueban. En la tabla 7 se muestran las dos combinaciones de métodos con las cuales los autores lograron una exactitud máxima en *Movie Fights* o *Surveillance Camera Fight*. Como se observa los métodos funcionan con diferente exactitud en los conjuntos de datos, obteniendo mejores datos en uno que en otro. La exactitud con nuestro método es similar, con un promedio de 67%. Con *Movie Fight* se obtuvo una exactitud de 95%, que es ligeramente menor a la que obtuvieron con otra red convolucional llamada *Xception* y *Bi-LSTM* con atención.

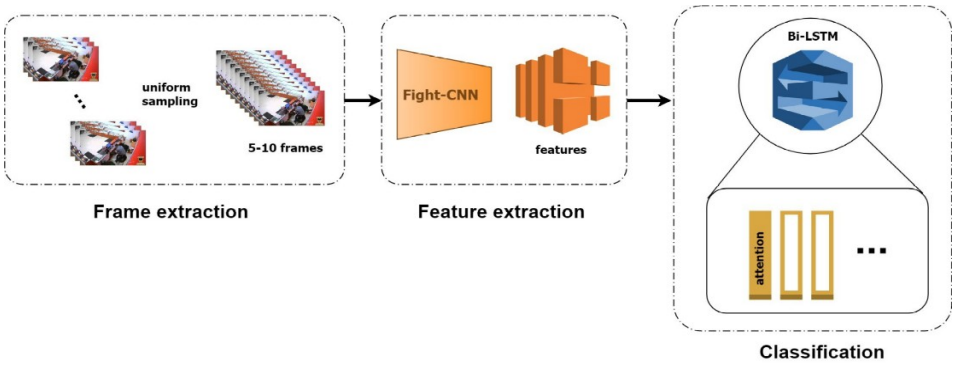


Figura 25: Método con Red convolucional Fight y Bi-LSTM con atención [9]

Tabla 7: Comparación con Akti et al. [9]

Extracción de características	Clasificador	Exactitud con *Movie Fights Dataset*	Exactitud con *Surveillance Camera Fight Dataset*
Xception	*Bi-LSTM*	98%	63%
	Bi-LSTM + atención	**100%**	69%
Fight-CNN	*Bi-LSTM*	80%	70%
	Bi-LSTM + atención	90%	**72%**
Open Pose	*Bi-LSTM*	95%	67%

El otro método con mayor exactitud de clasificación en *Movie Fights* fue el creado por Gracia *et al.* [10], donde utilizan zonas de alto movimiento como característica para reconocer acciones, usando tres clasificadores (*SVM*, Bosques aleatorios y AdaBoost). Obtienen los mejores resultados para este conjunto usando Bosques Aleatorios con su método, sin embargo, con los otros métodos obtienen menor exactitud.

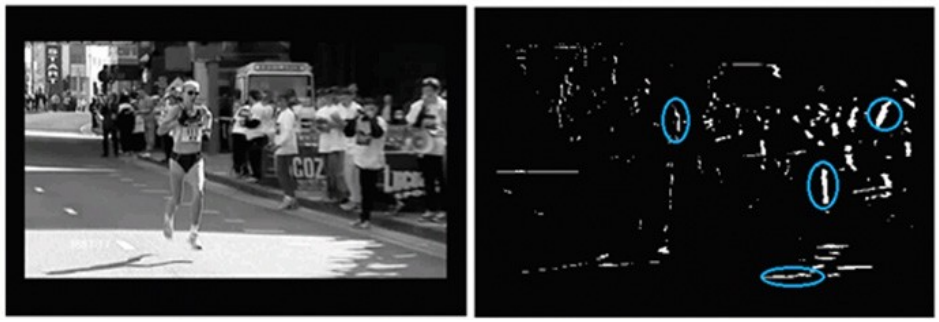


Figura 26: Extracción de motion blobs.

Tabla 8: Comparación con Gracia et al. [10]

Extracción de características	Clasificador	Exactitud con *Movie Fights Dataset*
Motion Blobs	*SVM*	87%
	AdaBoost	82%
	RF	98%
Open Pose	*SVM*	91%
	AdaBoost	92%
	RF	82%
	Bi-LSTM	95%

El *Violence Detection Dataset* es de creación y publicación reciente, por lo cual carece aún de trabajos con los cuales comparar; con éste se obtuvo una exactitud de clasificación de 81%.

Tabla 9: Comparación con trabajos relacionados.

Método	*Movie Fights*	*Surveillance Fights*	*Violence Detection*
Motion SIFT y *Histogram Intersection Kernel [12]*	90%	-	-
Red convolucional *Fight* y *Bi-LSTM* [9]	80%	**70%**	-
Motion Blobs y Bosques Aleatorios [10]	98%	-	-
Red convolucional Xception y *Bi-LSTM* [9]	**98%**	63%	-
Open Pose y *Bi-LSTM*	95%	67%	81%

Para aproximar el tiempo que toma entrenar y evaluar un conjunto, se realizó una ejecución de las diferentes partes del proceso con el conjunto *Movie Fights*. Cada etapa representa un programa con una función específica. Se midió el tiempo de ejecución en una computadora con CPU Intel Core i3-1005G1, de 1.20GHz × 4, por medio de Python 3.8 y las librerías previamente mencionadas.

El proceso funciona de la siguiente forma: Un primer programa extrae cuadros de los videos, los procesa y ejecuta *Open Pose* para cada imagen, obteniendo 10 matrices de personas por cada video. Un segundo programa calcula los ángulos de partes del cuerpo, hace el escalado y los transforma en vectores. El último programa divide el conjunto de vectores para entrenamiento y evaluación, y ejecuta el clasificador *Bi-LSTM.* En la tabla siguiente se presentan tiempos aproximados de cada etapa para el conjunto completo.

Tabla 10: Etapas del método.

Programa	Tiempo aproximado
Extracción de cuadros, preprocesamiento y ejecución de Open Pose (200 secuencias)	200 minutos. 60 segundos por video
Cálculo de matrices de ángulos y escalado (200 secuencias)	4 segundos
Entrenamiento de *Bi-LSTM* (160 secuencias)	10 segundos
Evaluación de *Bi-LSTM* entrenado (40 secuencias)	0.7 segundos

Como se puede observar, el mayor tiempo se utiliza para convertir las imágenes en matrices de posturas del cuerpo. El resto del proceso se puede completar en pocos segundos, debido al uso de los vectores de ángulos. Con un método integrado y entrenado, se podría clasificar un video en un minuto, aproximadamente, con variaciones según los parámetros que se usen y la optimización que se realice.

Capítulo 6 Conclusiones

Tras un estudio de las investigaciones de estado del arte en el área de detección de peleas, se desarrolló una propuesta para un método de clasificación. Como se mostró en la tabla 9 del capítulo anterior, el método propuesto en este proyecto tiene un desempeño comparable con trabajos del estado del arte, en cuanto a la exactitud que se obtiene.

Una de las fortalezas yace en la representación vectorial que se diseñó para transformar la información dada por *Open Pose*, que a su vez es una abstracción de las posiciones de las personas; esto permitirá adaptar y extender el uso de esta representación para más aplicaciones de reconocimiento de acciones. Igualmente puede ser modificado alguno de los módulos, ya sea el método de extracción de posturas o la clasificación, dado que el método solo es dependiente de las coordenadas de las personas y los ángulos y no de la técnica usada para extraerlos. Otra ventaja es el tamaño en memoria, dado que las acciones generales de múltiples personas en videos se lograron representar con secuencias de vectores de longitud fija. El proyecto se implementó con métodos y librerías de acceso abierto, para su posible replicación y extensión. La optimización de la complejidad computacional y el tiempo de ejecución pueden ser temas para trabajo futuro. Este método puede ser mejorado por medio de experimentos adicionales; se pueden realizar variaciones en el método de preprocesamiento, el número de cuadros extraídos, la cantidad de ángulos para los vectores de frecuencia; igualmente se pueden añadir funciones y cambiar la configuración de parámetros del clasificador *Bi-LSTM* para mejorar la clasificación.

El trabajo que se puede realizar en un futuro consistiría en mejorar aspectos del método, como la calibración de los parámetros del clasificador, optimizar los filtros para mejorar la calidad del video o usar diferentes técnicas, y hacer mejoras en las redes neuronales utilizadas. También se puede realizar la integración del método en un sistema completo que realice el proceso completo de detección de peleas, para su uso en aplicaciones reales.

Bibliografía

1. (2020) Centro de Comando, Control, Cómputo, Comunicaciones y Contacto Ciudadano. Recuperado de: https://www.c5.cdmx.gob.mx/. Fecha de acceso: 11 de mayo de 2021
2. Gobierno de la Ciudad de México. *Portal de Datos de la Ciudad de México*. https://datos.cdmx.gob.mx/pages/home/. Fecha de acceso: 11 de mayo de 2021
3. Tripathi, R. K., Jalal, A. S., & Agrawal, S. C. (2018). Suspicious human activity recognition: a review. *Artificial Intelligence Review*, 50(2), 283-339.
4. Nguyen, Duc Thanh, *Human detection from images and videos*, Doctor of Philosophy Thesis, School of Computer Science and Software Engineering, University of Wollongong, 2012. https://ro.uow.edu.au/theses/3665/. Fecha de acceso: 11 de mayo de 2021
5. Cao, Z., Hidalgo, G., Simon, T., Wei, S. E., & Sheikh, Y. (2018). OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. *arXiv preprint* arXiv:1812.08008.
6. Seemanthini, K., & Manjunath, S. S. (2018). Human detection and tracking using HOG for action recognition. *Procedia computer science*, 132, 1317-1326.
7. Naik, A. J., y Gopalakrishna, M. T. (2017). Violence Detection in Surveillance Video-A survey. *International Journal of Latest Research in Engineering and Technology* (IJLRET)., 11-17.
8. Gao, Y., Liu, H., Sun, X., Wang, C., & Liu, Y. (2016). Violence detection using oriented violent flows. *Image and vision computing*, 48, 37-41.
9. Aktı, Ş., Tataroğlu, G. A., & Ekenel, H. K. (2019). Vision-based Fight Detection from Surveillance Cameras. In 2019 Ninth International Conference on Image Processing Theory, Tools and Applications (IPTA) (pp. 1-6). IEEE
10. Gracia, I. S., Suarez, O. D., Garcia, G. B., & Kim, T. K. (2015). Fast fight detection. *PloS one*, 10(4), e0120448.
11. Ullah, A., Ahmad, J., Muhammad, K., Sajjad, M., & Baik, S. W. (2017). Action recognition in video sequences using deep bi-directional LSTM with CNN features. IEEE Access, 6, 1155-1166.
12. Nievas, E. B., Suarez, O. D., García, G. B., & Sukthankar, R. (2011). Violence detection in video using computer vision techniques. In *International conference on Computer analysis of images and patterns* (pp. 332-339). Springer, Berlin, Heidelberg.

13. Young, I.T., Gerbrands, J.J., van Vliet, L.J.(2007) Fundamentals of Image Processing. Delft University of Technology. Version 2.3. Recuperado de: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/TUDELFT/FIP2_3.pdf. Fecha de acceso: 11 de mayo de 2021
14. Gopinath, R., Ravi, A., Churiwala, S. (2019) An introduction to machine learning. doi: https://doi.org/10.1007/978-3-030-15729-6. Fecha de acceso: 11 de mayo de 2021
15. Gupta, Vikas. Multi-Person Pose Estimation in OpenCV using OpenPose. https://www.learnopencv.com/multi-person-pose-estimation-in-opencv-using-openpose/. Fecha de acceso: 11 de mayo de2021
16. Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences, 55(1), 119-139.
17. Domingo, C., & Watanabe, O. (2000, June). MadaBoost: A modification of AdaBoost. In *COLT* (pp. 180-189).
18. Olah, C. (2015) Understanding LSTM Networks. Recuperado de: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Fecha de acceso: 11 de mayo de 2021
19. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
20. CMU Perceptual Computing Lab. Open Pose: https://github.com/CMU-Perceptual-Computing-Lab/openpose/. Fecha de acceso: 11 de mayo de 2021
21. Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.
22. Abadi, M *et al.*. (2015) TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.
23. Pedregosa, F *et al.*. (2011) Scikit-learn: Machine Learning in Python, , JMLR 12, pp. 2825-2830.
24. Nievas, E. B., Suarez, O. D., García, G. B., & Sukthankar, R. (2011) Movie Fight Detection Dataset. Obtenido de: http://visilab.etsii.uclm.es/personas/oscar/FightDetection/
25. Ş. Aktı, G.A. Tataroğlu, H.K. Ekenel. Surveillance Camera Fight Dataset. Obtenido de: https://github.com/sayibet/fight-detection-surv-dataset. Fecha de acceso: 11 de mayo de 2021

26. M. Bianculli, N. Falcionelli, P. Sernani, S. Tomassini, P. Contardo, M. Lombardi, A.F. Dragoni, A dataset for automatic violence detection in videos, Data in Brief 33 (2020). doi:10.1016/j.dib.2020.106587.