APRENDIZAJE PROFUNDO PARA LA DETECCIÓN DE CAÍDAS EN PERSONAS VULNERABLES

Realizado por:
MARÍA PAULA SAA BELTRÁN

Universidad Santo Tomás
Facultad de Ingeniería Electrónica
Bogotá, Colombia
2022

APRENDIZAJE PROFUNDO PARA LA DETECCIÓN DE CAÍDAS EN PERSONAS VULNERABLES

Realizado por:
MARÍA PAULA SAA BELTRÁN

Proyecto de grado presentado como requisito para optar al título de:
INGENIERO ELECTRÓNICO

Dirigido por:
MSc. Gerson David Cruz Capador
Co-Dirigido por:
PhD. José Guillermo Guarnizo Marín

Grupo de Investigación GED (Grupo de Estudio y Desarrollo en Robótica)
Universidad Santo Tomás
Facultad de Ingeniería Electrónica
Bogotá, Colombia
2022

Autoridades de la Universidad

Rector General
Fray José Gabriel Mesa Angulo

Vicerrector Administrativo Y Financiero General
Fray Wilson Fernando Mendoza Rivera

Vicerrector Académico General
Fray Eduardo González Gil

Secretario General
Fray Mauricio Cortés Gallego

Decano División De Ingenierías
Fray Érico Juan Macchi Céspedes

Secretaria De División
Luz Patricia Rocha Caicedo

Decano Facultad De Ingeniería Electrónica
Carlos Enrique Montenegro Narváez

Nota de Aceptación

El trabajo de grado titulado **«Aprendizaje profundo para la detección de caídas en personas vulnerables»** realizado por la estudiante **María Paula Saa Beltrán** cumple con los requisitos exigidos por la **Universidad Santo Tomás** para optar al título de **Ingeniero Electrónico**.

Gerson David Cruz Capador: _____

José Guillermo Guarnizo Marín: _____

Armando Mateus Rojas: _____

Jaime Vitola Oyaga: _____

Bogotá D.C. _____ de 2022.

Advertencia

La Universidad Santo Tomás no se hace responsable de las opiniones y conceptos expresados en el trabajo de grado, solo velará por qué no se publique nada contrario al dogma ni a la moral católica y porque el trabajo no tenga ataques personales y únicamente se vea el anhelo de buscar la verdad científica.

Capítulo III –Art. 46 del Reglamento de la Universidad Santo Tomás.

Dedicatoria

Dedico este proyecto de tesis a mi familia, quienes siempre han confiado en mi y me han acompañado en cada paso de mi vida, me han permitido soñar y lograr cada una de las cosas que me he propuesto, sin ellos nada de esto sería posible.

Agradecimientos

Agradezco a Dios, a mi padres, a mis abuelitas, a mis directores, a mis docentes y compañeros, por permitirme lograr que este proyecto se realizara de la mejor manera posible, agradezco el apoyo, el tiempo, el conocimiento y la paciencia que me brindaron de manera constante y en todo momento, en especial, en la ejecución de mi proyecto de tesis.

Resumen

El proyecto de tesis, muestra el diseño, implementación y desarrollo de un algoritmo de aprendizaje profundo que permite realizar detección de caídas en especial en personal mayores que se encuentran viviendo solas, en espacios médicos o en centros de cuidados geriátricos.

Se busca que este proyecto pueda realizar esta detección, sin la necesidad de emplear dispositivos corporales que pueden generar inconvenientes en los pacientes, razón por la cuál se opta por emplear un algoritmo basado en redes neuronales recurrentes de tipo LSTM (Memoria prolongada de corto plazo), que tienen la capacidad de recordar información relevante en secuencias y preservarlo por varios instantes de tiempo.

Se realizan las pruebas en ambientes controlados, junto con personas que emulen las caídas y que no cuenten con ningún inconveniente de salud, la evaluación del proyectó se realiza a través de distintas métricas y pruebas en tiempo real.

Abstract

The undergraduate project shows the design, implementation and development of a deep learning algorithm that allows fall detection, especially in older people who are living alone, in medical centers or in geriatric care centers.

It is sought that this project can carry out this detection, without the need to use body devices that can generate inconveniences in patients, that is why it is chosen to use an algorithm based on recurrent neural networks of the LSTM type (Long short-term memory), that have the ability to remember relevant information in sequences and preserve it for several instants of time.

The tests are carried out in controlled environments, with people who emulate the falls and do not have any health issues, the evaluation of the project is carried out through different metrics and tests in real time.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Las caídas son situaciones de riesgo que afectan a la población mayor, dado que se suelen presentar con bastante frecuencia y el impacto que pueden generar en las personas puede afectar su salud en general y evitar que cuenten con buen desempeño en el desarrollo de sus actividades diarias. Gracias a la tecnología es posible que se puedan detectar este tipo de caídas, ofreciendo a las personas ambientes seguros y tranquilos.

Para el caso de este proyecto de grado, se busca implementar un algoritmo que permita realizar una detección adecuada, sin la necesidad de emplear dispositivos invasivos que puedan resultar incómodos para las personas que lo requieren, para ello se opta por emplear un algoritmo de aprendizaje profundo que tenga la capacidad de aprender cuándo una persona sufre una caída y posterior a ello detectarla.

Para la elaboración de este proyecto se opta por emplear una metodología que consta de cinco pasos, donde en los repositorios de la universidad se investiga a detalle todo lo relacionado con caídas y redes neuronales, luego de ello es necesario realizar una caracterización de todos los componentes a emplear en el proyecto, una vez se realiza esto se generan los conjuntos de datos que permiten se realice el adecuado entrenamiento de la red, posterior a ello se determina el tipo de red neuronal que mejor funcione para hacer detecciones en tiempo real y se realiza la implementación, para finalmente realizar las pruebas finales con el fin de validar que el algoritmo funcione adecuadamente.

La idea del proyecto no es solo contribuir al cuidado de pacientes que se encuentren en hospitales, hogares geriátricos y hogares sin supervisión médica o de familiares, se busca también apoyar a las personas que tengan algún tipo de discapacidad y requieran atención y supervisión a lo largo del día, todo esto con el fin de prevenir inconvenientes que pueden generar repercusiones a largo plazo que pueden afectar a los pacientes.

Capítulo 2

Planteamiento del problema

Las caídas constituyen uno de los síndromes geriátricos más importantes por su alta incidencia y por la elevada morbimortalidad

La sociedad colombiana actual, cuenta con diversas comunidades vulnerables, un grupo representativo son los adultos mayores, 7'412.407 personas pertenecen a este grupo de acuerdo a diversos estudios demográficos realizados por el DANE para el año 2020 [13]. Las caídas son uno de los eventos más peligrosos para la vida de este grupo de personas, por su alta incidencia [51], estas ocurren por diferentes inconvenientes asociados a distintos factores, que una vez identificados logran ser controlados. Los factores de accidentalidad pueden ser clasificados en tres dimensiones distintas. La primera se encuentra relacionada con la salud de la persona, como los son las enfermedades crónicas, problemas en las funciones neuronales o cardiovasculares, problemas cognitivos o asociados a caídas. La segunda hace referencia al comportamiento de la persona, como el estilo de vida en cuestiones de consumo de sustancias o alcohol en exceso, y la tercera se encuentra asociada a factores del entorno, ya sean externos o internos que generan peligros [9]. Es importante recalcar que muchas personas que son adultas mayores tienen discapacidad, cosa que ya de por sí, es un inconveniente enorme, motivo por lo cual es necesario que las que dicha población, tenga supervisión casi en la totalidad del día.

La detección automática de caídas puede reducir el tiempo de acceso a los centros médicos y por consiguiente el riesgo inminente de deceso, las diferentes caídas que presentan las personas no solo causan daño físico, sino que también generan daños psicológicos, médicos e incluso sociales. Es por eso que es indispensable generar sensores de monitoreo automático o detectores de caídas para realizar avisos a las personas que se encuentran al tanto de adultos mayores con el fin de brindar una colaboración oportuna al momento de presentarse una caída [20].

Existe gran cantidad de tecnologías que tienen la capacidad de detectar caídas, en muchos casos se emplean dispositivos que se conectan al cuerpo y emiten señales por medio de monitores [41], estos dispositivos, suelen ser incómodos para las personas que deben emplearlos e inclusive pueden impedir algún tipo de movimiento, ya que se deben conectar en lugares específicos del cuerpo, como lo son el cuello o la parte superior de las piernas, muchas veces estos dispositivos, se pueden caer, generando mayor incomodidad en las personas que lo emplean.

A partir de la problemática planteada, se presenta la siguiente pregunta de investigación:

¿Qué nivel de confiabilidad presenta un algoritmo basado en el aprendizaje profundo, que tenga

la capacidad de detectar caídas en personas vulnerables, con el fin de generar una atención oportuna a un paciente en riesgo?

Capítulo 3

Antecedentes.

A continuación se presenta la revisión del estado del arte, relacionado con todas las temáticas de mayor relevancia en el proyecto.

3.0.1. Caídas en personas mayores.

Las caídas se pueden definir como una consecuencia de algún tipo de suceso que precipita al individuo al suelo, en contra de su voluntad, las caídas son un síntoma que se tiende a asociar con una elevada morbimortalidad en los adultos mayores, muchas veces se suelen asociar las caídas como algún tipo de síntoma de una enfermedad o un trastorno, razón por la cual, no necesariamente se debe considerar que una caída se puede generar por razones ambientales o de edad, cabe recalcar que las caídas se tienden a generar en su mayoría en personas mayores ya que los cambios propios de envejecimiento pueden llegar a contribuir en que estos inconvenientes se presenten [22].

Las caídas no solo pueden generar consecuencias físicas, sociales y psicológicas también requiren atención médica. El hogar y las vías públicas son considerados los lugares que mayor cantidad de eventos pueden generar, por otro lado, las estructuras corporales que tienden a afectarse con mayor facilidad con la cadera, la cabeza, la cara, las manos, el hombro y el tobillo [17]. Es importante tener presente que las caídas pueden ocurrir en cualquier edad, pero las consecuencias no son necesariamente las mismas, ya que, en personas de mayor edad, el riesgo de discapacidad o mortalidad, se eleva de manera considerable [51].

3.0.1.1. Factores de riesgo.

Estos pueden ser clasificados en dos grupos diferentes, los factores intrínsecos que son propios del paciente y son determinados por los cambios a nivel fisiológico generados por la edad y los factores extrínsecos, los cuales hacen referencia a factores ambientales, a entornos arquitectónicos o elementos de uso personal del paciente.

Los factores intrínsecos que generan una caída se pueden generar por causas neurológicas, como lo son los accidentes cerebro vasculares, las mielopatías, la enfermedad del Parkinson, las convulsiones, la demencia, entre otras patologías, también se pueden generar por causas cardiovasculares, como lo es el infarto en el miocardio, las arritmias cardiacas, las embolias pulmonares, la hipertensión arterial entre otros, de igual manera de dan por causas músculo esqueléticas, como los son las deformidades en la columna vertebral, la artrosis, la artritis, la miositis, entre otras patologías,

existe otro grupo de causas donde se encuentran las intoxicaciones, los sincopes neurovegetativos, la depresión, la ansiedad, la hipoglicemia, la incontinencia urinaria entre otros, por otro lado el consumo de ciertos medicamentos puede generar en los pacientes efectos secundarios, que pueden ocasionar somnolencia, alteraciones visuales, pérdida de reflejos, entre otros, razón por la cual es posible que se pueda generar una caída [17].

Los factores extrínsecos que producen una caída se pueden generar por barreras arquitectónicas del hogar, como lo son los muebles mal ubicados, duchas resbaladizas, estantes elevados, obstáculos en los suelos, camas altas, entre otros, de igual manera se pueden generar las caídas por costumbres peligrosas que pueden realizar los pacientes, como caminar sin zapatos, subirse en andamios, realizar movimientos excesivamente bruscos con el cuerpo, abusar del alcohol, realizar esfuerzos físicos exagerados, entre otros [51].

3.0.1.2. Consecuencias de las caídas.

Existen gran cantidad de consecuencias producto de las caídas, se destacan las siguientes [17]:

- **Lesiones superficiales:** Hacen referencia a caídas, arañazos o erosiones que afectan el tejido más superficial de la piel.
- **Contusiones simples:** Se generan por golpes y pueden producir enrojecimientos en la piel o eritema, sin necesidad que generar algún tipo de alteraciones o desgarros en los planos profundos.
- **Equimosis:** Hace referencia a una hemorragia superficial que se produce por lesiones subcutáneas.
- **Laceración superficial o profunda:** Se produce la abertura o desgarro de la piel producto de una lesión.
- **Luxación:** Se genera cuando hay una separación de dos huesos en el lugar donde existe una articulación.
- **Fractura:** Es la ruptura de un hueso, en este caso se produce una separación de los extremos de los huesos.
- **Caídas mortales.** Como su nombre lo indica, las personas pueden sufrir caídas de tal magnitud que su vida puede estar en riesgo.

3.0.2. Proyectos implementados para la detección de caídas.

El reconocimiento de la actividad humana juega un papel crucial en las interacciones de humanos y computadoras, logrando generar dispositivos capaces de vigilar la salud de las personas de formas invasivas mediante dispositivos conectados en alguna parte del cuerpo humano y no invasivas mediante dispositivos remotos como cámaras o sensores, para el caso de las interacciones no invasivas existen diversas formas de adquirir información, un método utilizado se realiza a partir de tomas de vídeo mediante cámaras de alta resolución, las cuales logran la adquisión de conjuntos de datos, que son preprocesados por medio de diversos algoritmos y son entrenados con el fin de detectar alguna tarea en particular [30].

3.0.2.1. Proyectos implementados con tecnologías invasivas

Los proyectos relacionados con la generación de dispositivos para la vigilancia muchas veces suelen manejar tecnologías invasivas, donde sensores o algún tipo de dispositivo en particular es conectado en la persona, para emitir señales con el fin de obtener un conjunto de datos en específico, uno de ellos es "Validation of 24-hour ambulatory gait assessment in Parkinson's disease with simultaneous video observation" que consta de 4 cámaras y un monitor de marcha ubicado en encima del cuello del paciente, que básicamente es un giroscopio de doble eje, con esto se logra identificar si el paciente se encuentra caminando o acostado y con ayuda de las cámaras se observa el tipo de actividades locomotoras que la persona está desarrollando y se logran captar en el giroscopio [41].

Otro proyecto que presenta tecnología invasiva pero que logra evidenciar accidentes es "Reliable Fall Detection System Using an 3-DOF Accelerometer and Cascade Posture Recognitions", que consta de un acelerómetro de 3 grados de libertad ubicado en la cintura del paciente, al momento de presentarse una caída el sistema implementado en el microcontrolador determina si se trata de una caída real, de acuerdo a los datos obtenidos en los 3 ejes del acelerómetro [19]. Existen proyectos que realizan la manipulación de datos a través de redes neuronales profundas, pero que de igual forma utilizan algún tipo de tecnología invasiva, como el proyecto titulado, "Deep Neural Network–Based Double-Check Method for Fall Detection Using IMU-L Sensor and RGB Camera Data", la idea de emplear un acelerómetro es que algunos detectores pueden generar falsas alarmas, pero gracias al sensor IMU-L que es la combinación de un acelerómetro y un giroscopio y a un sensor de señales, es posible realizar la detección de manera más acertada, junto a la detección de datos a través de un sensor RGB conectado a un robot, la idea es que todo los datos puedan ser entrenados por medio de una red neuronal recurrente, con el fin de detectar si una persona mayor tuvo o no una caída [29].

3.0.2.2. Proyectos implementados con tecnologías no invasivas

En el caso de los proyectos no invasivos, para detectar un evento inusual por medio de video vigilancia, es menester generar un algoritmo que permite detectar el proceso de manera eficiente, por lo cual en el artículo "Automated Unusual Event Detection In Video Surveillance", se presenta un conjunto de pasos necesario para este proceso, primero se genera el vídeo, se realiza la extracción del fondo y se realiza la operación de detección morfológica en este caso es necesario detectar movimientos de manera eficaz, lo cual permite delimitar los datos obtenidos que posteriormente se clasifican con el fin de extraer características relevantes, en este caso primero es necesario realizar la verificación de la posición de la persona de acuerdo a sus características morfológicas, luego es preciso verificar la silueta y sus respectivos patrones de color y finalmente es necesario tener presente el ángulo de caída, para luego proceder a la clasificación de las imágenes [10]. De igual forma se puede realizar la detección a través de radar tipo doopler, como en el monitor de caídas titulado "Elderly Fall Detection With Vital Signs Monitoring Using CW Doppler Radar", en este caso se emplea un radar doppler de onda continua de 24 GHz, ya que tiene una alta capacidad de detectar movimientos humanos y no se afecta por condiciones mínimas, la idea de este proyecto es ubicar 2 radares en un espacio controlado y a través de las vibraciones que se captan en el radar, se logré identificar si ha ocurrido o no una caída [28].

3.0.2.3. Proyectos implementados con Redes Neuronales Convoluciones, Redes Neuronales Recurrentes y Redes Neuronales Recurrentes de tipo LSTM

Por otro lado, existen algoritmos basados en distintos tipos de redes neuronales para la detección de caídas, como lo son las redes neuronales convolucionales (CNNs) y las redes neuronales recurrentes (RNNs), de igual manera existe un tipo de red neuronal recurrente denominada LSTM que también se ha empleado para la detección de caída.

Para el caso del primer proyecto se emplean las redes neuronales convolucionales profundas, dicho trabajo fue presentado en el Décimo Congreso Internacional de Procesamiento de Imágenes y Señales, Ingeniería biomédica e Informática en el año 2017, denominado "Fall Detection for Elderly Person Care Using Convolutional Neural Networks", en este caso este tipo de redes, facilitan el aprendizaje a través de datos representativos, se aplican directamente a cada cuadro de imagen de los vídeos logrando aprender algunas características en la forma de los seres humanos y logrando con esto determinar si se presenta una caída [37] . Otro trabajo representativo basado en redes neuronales convolucionales, fue presentado en la conferencia internacional de Ciencia de Datos y sus aplicaciones, titulada "Real-time Falling Detection System for Elderly using CNN", la idea de implementar este proyecto es alertar sobre la caída ocurrida a un adulto mayor y generar una alerta, esto se logra gracias a la implementación de un monitor de vídeo que utiliza una red neuronal convolucional, compuesta de diferentes capas convolucionales que entrenan un Dataset de 220 cuadros de diferentes vídeos, que permite realizar la extracción de características relevantes para la detección de caídas, logrando un acierto de más del 90 % [25]. Por otro lado, se evidencia otro proyecto significativo, titulado, "A deep neural network for real-time detection of falling humans in naturally occurring scenes", que, a partir de vídeos y posteriores imágenes dinámicas, realiza la detección de las caídas a través de un modelo de red neuronal preentrenado (VGG-16) y a partir de las predicciones obtenidas en las métricas determina si existe algún tipo de caída o no [24].

Por otra parte para las redes neuronales recurrentes se presenta un trabajo denominado "Lightweight Real-time Fall Detection using Bidirectional Recurrent Neural Network", que maneja redes de carácter bidireccional, que básicamente busca entrenar todos los datos de entrada en un tiempo pasado o futuro de manera simultánea; empleando este tipo de arquitectura, se logra predecir las caídas identificando información significativa basada en secuencias [34]. Otro artículo relevante es "Deep Recurrent Neural Networks for Human Activity Recognition", explica la importancia del uso del lenguaje profundo en el reconocimiento de la actividad humana, al implementar redes recurrentes el sistema es capaz de capturar algún tipo de dependencia de largo alcance en secuencias de entradas con longitudes variables, es por eso que se generan estas redes basadas en memorias largas a corto plazo, ya que se modelan secuencias y se reemplazan los nodos con celdas de memoria que poseen recurrencia externa e interna [43].

De igual manera se presentan proyectos basados en las redes de tipo LSTM para la detección de caídas, como el proyecto "Human action recognition based on spatial–temporal relational model and LSTM-CNN framework", que a través de la detección del cuerpo humano proponen realizar las detecciones a través de las combinaciones de las redes neuronales convolucionales profundas y de redes de tipo LSTM bidireccionales, se obtienen los puntos de referencia del cuerpo humano y con ayuda de estos datos se realiza la implementación una red convolucional retroalimentada que a la salida se conecta con la red bidireccional con el fin de realizar la detecciones de las acciones [53]. Otro proyecto significativo, fue presentado en el Décimo Noveno Congreso Internacional de

Ingeniería Eléctrica/Electrónica, Informática, Telecomunicaciones y Tecnologías de la Información (ECTI-CON), denominado "Fall Detection for The Elderly using YOLOv4 and LSTM", este proyecto emplea el algoritmo YOLOv4 para la detección de posturas que se basa en una red neuronal convolucional y la red de tipo LSTM para la detección de las secuencias de estas, una gran ventaja de este proyecto es la carencia de implementaciones de proyectos unificando ambos tipos de redes y por otro lado, el manejo de un ambiente no controlado para la detección de las diferentes posturas [11].

Asimismo, se evidencian otro tipo de proyectos como "RGB-D Fall Detection via Deep Residual Convolutional LSTM Networks", que genera un modelo integrable para una detección eficiente a partir de imágenes obtenidas a través de un sensor Kinect RGB-D, con una arquitectura de aprendizaje profundo que contiene redes convolucionales y recurrentes que permiten la detección de eventos, al igual que el proyecto anterior, se busca que primero se haga una detección de características y posterior a ello una detección de secuencias a partir de imágenes de profundidad [1]. El proyecto presentando en el Congreso Internacional de Robótica, Automatización, Inteligencia Artificial e Internet de las Cosas (RAAICON), titulado, "Robust Pose-Based Human Fall Detection Using Recurrent Neural Network", basado en un modelo de red neuronal recurrentes con una arquitectura de tipo LSTM, con esto se busca omitir la apariencia de las personas, al igual que la información del ambiente, con el fin de solo evidenciar los patrones de caídas, esto se realiza a través de los puntos de referencia del cuerpo que se toman como entradas de la red neuronal tipo LSTM de dos capas [29].

Existen diversos proyectos relacionados con el comportamiento del cuerpo de los seres humanos y que tienen como fin la detección de posiciones o estimación de posturas como lo es el proyecto "Bed-Exit Prediction Applying Neural Network Combining Bed Position Detection and Patient Posture Estimation", el cual busca mediante el análisis de diversas imágenes y una cámara monocular detectar cuando una persona se encuentra sentada en el borde de una cama con riesgo a caerse, logrando de esta forma reducir relativamente el riesgo de caídas, para este caso se propone detectar la parte superior de un paciente y los cuatro lados de la cama, con ayuda de una red neuronal profunda logra detectar la posición específica de la persona, es decir, si está sentado, en posición de dormir o incluso a punto de caerse, esto se logra realización experimentación con base en la detección de datos y posterior aprendizaje de las redes [31].

3.0.2.4. Proyectos implementados con Internet de las Cosas y aprendizaje profundo

Como consecuencia de los proyectos anteriores, de diversos estudios e innovación, se han realizado propuestas de detección de caídas integrando el Internet de las Cosas (IoT), pero más específicamente el Internet de las Cosas de la Salud (IoHT), que es básicamente describe dispositivos conectados a internet que permiten la comunicación, pero enfocados en la parte médica. Un proyecto representativo es "Deep-Learning-Enhanced Human Activity Recognition for Internet of Healthcare Things", tiene como finalidad el aprendizaje profundo mejorado para el reconocimiento de la actividad humana en entornos del Internet de las Cosas de la Salud [62]. Con la implementación de este tipo de tecnologías es posible crear algoritmos lo suficientemente elaborados para detectar patrones que permitan reconocer actividades específicas de las personas, con el fin de monitorear su salud en todo momento y cuando se presente algún evento desafortunado generar llamados de manera remota.

Capítulo 4

Justificación

De acuerdo a lo expuesto en el planteamiento del problema, es evidente la necesidad de crear dispositivos capaces de monitorear a las personas, cuando sufren algún percance de salud, gracias al monitoreo es mucho más fácil tener supervisión y cuidado para las personas. Los accidentes son sucesos muchas veces inevitables, pero en algún punto pueden ser manejados de manera apropiada, ya que, al ser detectados, se toman acciones apropiadas para que el paciente pueda continuar con sus actividades cotidianas de manera normal [18].

En su mayoría las caídas que tienden a tener las personas mayores por mínimas que sean, pueden llegar a afectar de manera profunda a la persona. En el aspecto físico una fractura o una lesión en los tejidos blandos puede producir falta de movilidad o movilidad reducida, que si no es tratada de manera rápida puede resultar en inmovilidad total en algunos casos, al igual que la dependencia total. Por otro lado, para el aspecto psicológico, se pierde confianza al momento de realizar cualquier actividad por mínima que parezca, ya que la persona vive con el miedo de sufrir un accidente similar. Por el lado de los aspectos sociales, la persona que presenta este tipo de afecciones tiene el temor de terminar en algún lugar geriátrico por el simple hecho de tener percances a nivel familiar o amistoso y finalmente para la parte médica y económica, muchas de las lesiones requieren gran cantidad de terapias, hospitalización y demás cosas que requieren gran cantidad de dinero [20].

Existen diversos factores por los cuales se producen los accidentes, estos pueden ser intrínsecos en las personas, como lo son la edad, el historial de caídas, los problemas médicos agudos o crónicos, como lo son enfermedades como el Parkinson, la osteoporosis, los problemas cardiovasculares y de igual manera la afectación de las funciones neuronales, mientras que por otro lado existen problemas externos que pueden fomentar los percances como es el entorno externo a la vivienda de la persona, calles irregulares o nieve en el suelo, lo anterior, es algo inherente por lo cual resulta complejo modificarlo. [33].

La razón de emplear reconocimiento de la actividad humana en el proyecto, tiene como finalidad la interacción que se puede generar entre las máquinas y las personas, en aplicaciones de la vida real como es el cuidado de la salud. Esta práctica presenta dos tipos de sistemas, los primeros basados en sensores o cámaras externos y los otros basados en sensores portátiles conectados al cuerpo humano, muchos de los patrones generan algún tipo de señal que permite la segmentación e identificación de una actividad en específico. Gracias al reconocimiento de la actividad humana se mejoran los métodos tradicionales de aprendizaje automático[43], ya que se logra determinar gran cantidad de características, ya sea logrando una abstracción en alto o en bajo nivel, todo

esto se consigue con ayuda del aprendizaje profundo que tiene la capacidad de extraer patrones e información relevante de las imágenes [43].

Mediante el uso de tecnologías como la visión computacional y el aprendizaje profundo, se busca generar sistemas que permitan la supervisión de las personas. Gracias al aprendizaje profundo es posible obtener datos altamente relevantes por medio de imágenes obtenidas por cámaras de vídeo, donde es necesario realizar abstracción de datos en alto nivel, generando mapas binarios, los cuales permiten entrenar una red neuronal [37] y con ello se logra que esta tenga la capacidad de detectar cualquier tipo de evento anómalo en las personas mayores.

De igual manera, en esta propuesta de proyecto se abarcan temas relacionados con la Misión Sabios 2019 propuesta por el Ministerio de Ciencia Tecnología e Innovación, en lo temas relacionados con "Tecnologías Convergentes – (Nano, Info y Cognotecnología) - Industrias 4.0", ya que se busca la implementación de tecnologías para la transformación social y en o relacionado con el tema de "Ciencias de la Vida y de la Salud", con el fin de mejorar la salud e igualdad entre las poblaciones del país [14].

Capítulo 5

Impacto social

Este proyecto tiene como finalidad contribuir al cuidado de los adultos mayores, con el fin de prevenir, supervisar e informar en caso de ocurrir algún incidente relacionado a una caída. Es evidente el inminente crecimiento en la tasa de accidentalidad en personas mayores ya que a lo largo de los años está población aumenta de manera paulatina [13], por el cual se busca reducir en gran medida este tipo de sucesos y en caso de que ocurra se pueda prestar una atención ágil y oportuna, para evitar percances graves de salud a largo plazo o incluso decesos indeseados. Con ayuda del algoritmo basado en aprendizaje profundo y mediante el reconocimiento de la actividad humana, es posible lograr esto, gracias a ello las personas que viven solas o se encuentran solas pueden estar en total tranquilidad, ya que en dado caso de presentarse algún inconveniente una alerta permite una atención rápida.

Se toma como precedente de igual manera la ley 1850 del 2017, la cual maneja medidas para la protección de los adultos mayores en Colombia, en su artículo 12, se genera un programa de asistencia para personas de la tercera edad [15]. Con el proyecto se busca contribuir en pequeña medida al cuidado de este grupo social, con el fin de que puedan tener una vejez grata y de en caso ser necesario tener asistencia continua a lo largo del día. Por otro lado en la Ley 2055 de 2020, se busca que las personas de edad tengan una vejez digna (artículo 6), un derecho a la independencia y a la autonomía (artículo 7), derecho a la seguridad y a una vida sin ningún tipo de violencia (artículo 9), derecho a recibir cuidados a largo plazo (artículo 12) y derecho a la salud (artículo 19), por medio del proyecto se logra generar un apoyo enorme a las personas de edad, permitiendo que ellas puedan vivir su vida sin complicaciones y recibiendo el apoyo necesario para tener una vida tranquila, previniendo problemas de salud que pueden generar un sinnúmero de secuelas graves [16].

Es importante tener en cuenta que este proyecto pese a que se ha planteado para la detección de caídas en personas mayores, puede ser empleado en individuos que no lo sean o presenten algún tipo de comorbilidad relacionada con enfermedades de base en la persona o con procesos de recuperación asociados al Covid-19 que requiera la atención de manera continua con el fin de evitar contratiempos graves en su salud y condición física. De igual manera puede ser empleado en diferentes centros de salud que presentan gran afluencia de pacientes, como lo son los centros oncológicos o centros renales, donde las personas pueden tener complicaciones durante el proceso de quimioterapia o diálisis y sus cuidadores se encuentran atendiendo otro tipo de actividades, o pacientes que igualmente requiere de atención oportuna.

Capítulo 6

Objetivos

6.1. General

Implementar un algoritmo de aprendizaje profundo, con el fin de detectar caída de personas, mediante vídeos obtenidos en tiempo real.

6.2. Específicos

1. Realizar una revisión del estado del arte, relacionado con la detección de eventos anómalos, adquisición de imágenes y aprendizaje profundo.

2. Caracterizar un sistema de vídeo que tenga la capacidad de detectar eventos tales como caídas en tiempo real.

3. Obtener el conjunto de datos que permitan el entrenamiento del sistema basado en aprendizaje profundo, para detectar las caídas y de igual manera adquirir las métricas que permitan la evaluación del sistema.

4. Identificar el tipo de sistema adecuado basado en aprendizaje profundo para la detección de eventos como caídas, a partir de información obtenida mediante vídeos en tiempo real, así como las métricas que permitan la evaluación del sistema a partir de la revisión del estado del arte.

5. Realizar pruebas con el sistema de visión y el algoritmo, con el fin de ajustar el sistema en su conjunto y efectuar la validación de este en las instalaciones del laboratorio de robótica de la Universidad Santo Tomás.

Capítulo 7

Marco teórico

7.1. Redes Neuronales

La implementación de las redes neuronales artificiales han sido el resultado de lograr que una máquina tenga funciones básicas como las del cerebro humano [3] y de igual forma que tenga la capacidad de emularlas, son un tipo de algoritmo de aprendizaje máquina, como se menciona anteriormente inspirado en el cerebro biológico y busca solucionar gran cantidad de problemas de forma bastante similar a como lo haría el cerebro humano [58]. Gracias a su aprendizaje adaptativo y auto organizado, logra tener gran tolerancia a los fallos, al igual que una buena operación en tiempo real, que permite que puedan ser empleadas en tecnologías existentes, gracias a ello, pueden ser utilizadas en gran cantidad de áreas como la parte de la salud, la parte militar, financiera e incluso industrial [48].

7.1.1. Redes Neuronales Biológicas

Para entender de mejor manera el funcionamiento de una Red Neuronal Artificial, es necesario tener claro el funcionamiento de una red neuronal biológica, que consta de tres partes principales: el cuerpo celular o soma, las dendritas y el axón. Las dendritas tienen la capacidad de recibir las señales o información de otras neuronas, por otro lado, el cuerpo de la célula permite la adquisición y procesamiento de dicha información y finalmente los axones envían las señales de salida a otras neuronas para continuar con el flujo de información [56], cabe recalcar que la unión de dos neuronas o perceptrones forman una red y es a través de esta red es que se realiza la trasferencia de información por medio de la sinapsis (impuso nervioso que se produce a través de las neuronas y que posibilita su comunicación [21]) entre las dendritas de una de las neuronas y el axón de la otra neurona [23].

7.1.2. Redes Neuronales Artificiales

Teniendo claro el funcionamiento de una red neuronal biológica, resulta sencillo analogar sus funciones con una red neuronal artificial, estás neuronas cuentan con una capa de entrada que es la encargada de recibir los datos, cada dato se conecta a través de líneas a las demás capas ocultas de la red neuronal que son denominados pesos para procesar las entradas de manera secuencial hasta llegar a la capa de salida, básicamente la entrada de la neurona actúa como una capa de entrada, mientras que la neurona en si es la capa oculta y finalmente la salida a la siguiente neurona hace

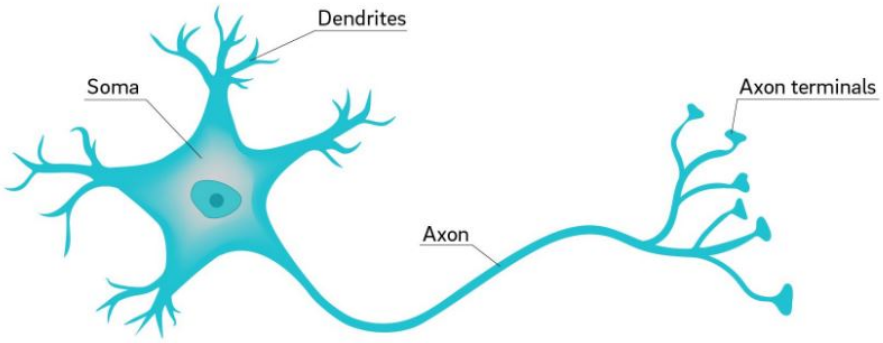
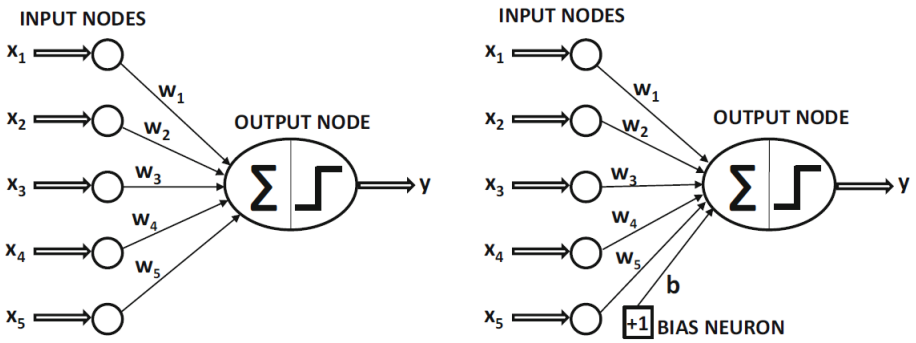


Figura 7.1: Red Neuronal Biológica [60]

referencia a la capa de salida [44]. Básicamente una red neuronal permite el procesamiento de la información multiplicando una variable de entrada, por un peso específico, cabe recalcar que dichas entradas son parámetros que la red debe aprender a lo largo del entrenamiento, una vez se realiza la multiplicación de ambos parámetros es necesario sumarlos y pasar estos datos a través de una función de activación, que se explica manera más detallada en la sección 7.1.2.3 , con el fin de normalizar dicha suma antes de que se genere una salida, como se evidencia en la figura 7.2.



(a) Perceptron without bias (b) Perceptron with bias

Figura 7.2: Arquitectura básica de un perceptrón [2]

Todas las redes neuronales, cumplen con los mismos requisitos en lo relacionado a entradas, pesos y uso de funciones de activación, es por ello que a partir de la ecuaciones 7.1 y 7.2, se puede tener una ecuación que permita representar de manera matemática el funcionamiento de cualquier red neuronal [8].

$$H = X1 * W1 + X2 * W2 + X3 * W3 + ... + Xn * Wn + B \tag{7.1}$$

$$Y = f(H) \tag{7.2}$$

Dónde:

- Y → es el valor de la red neuronal
- W → hace referencia a los pesos entre los nodos, las capas anteriores y la salida del nodo.
- X → representa los valores de los nodos de las capas previas

- B → hace referencia al sesgo, que representa al valor adicional presente en cada una de las neuronas, este valor es un peso sin ningún tipo de termino de entrada, tiene gran utilidad al permitir mayor capacidad de ajuste que no depende de las capas anteriores.
- H → es el valor del nodo intermedio.
- *f()* → es la función de activación.

Gracias a su increíble evolución y la facilidad de su estructura cuentan con ciertas ventajas [48]:

1. **Aprendizaje adaptativo:** Capacidad de una máquina para aprender a realizar un tipo de tarea determinado a partir de un entrenamiento o experiencia inicial.
2. **Auto organización:** Las redes neuronales artificiales tienen la capacidad de generar organizaciones o representaciones de información durante toda su etapa de aprendizaje.
3. **Tolerancia a los fallos:** Las RNA tienen la capacidad de manejar su información de manera distribuida, lo que permite que se permite que, al momento de ocurrir un daño en la red, no todas las capacidades se pierdan en su totalidad, sino que esta se pueda dañar de manera parcial.
4. **Capacidad de generalización:** Ante la entrada de nuevos resultados las redes tienen la capacidad de generar resultados coherentes basándose en el problema para el cual han sido entrenadas.
5. **Operación en tiempo real:** El computo de estas redes se puede ejecutar de manera paralela.

Las redes neuronales, también pueden ser denominadas como fijas, ya que los valores de los pesos permanecen estáticos o adaptativas que dados los valores de la red se adaptan y pueden cambiar, asimismo, las redes siempre deben tener 3 tipos de conjuntos de datos.

1. **Conjunto de datos de entrenamiento:** Se utilizan para ajustar los pesos de la red neuronal.
2. **Conjunto de datos de validación:** Se emplea para minimizar el problema de sobreajuste.
3. **Conjunto de datos de prueba:** Se utiliza como una prueba final para medir la precisión con la que la red ha sido entrenada.

7.1.2.1. Optimizadores

Los optimizadores tienen la capacidad de perfeccionar los parámetros que permiten reducir los errores cometidos por la red, realizando cambios tanto en los pesos como en las tasas de aprendizaje. A continuación, se muestran los tres optimizadores principales [52][57]:

- **Adagrad (Descenso de Gradiente Adaptativo):** Es un método adaptativo que permite ajustar la tasa de aprendizaje de acuerdo a los parámetros de la red, en este caso se realizan cambios más significativos en los parámetros no tienden a variar mucho y se realizan cambios menores para los valores que varían muy poco.

En la ecuación 7.3, se evidencia la fórmula de Adagrad que permite realizar las actualizaciones.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \varepsilon}} \odot g_t \tag{7.3}$$

Dónde: G_t es la suma de los cuadrados de todos los gradientes anteriores con todos los parámetros θ a lo largo de la diagonal, $\odot$ es el producto punto matriz-vector, g_t es el gradiente en el paso de tiempo t, η es la tasa de aprendizaje y ε es el termino suavizado que permite evitar que se realice una división por cero, en este caso su valor es del orden $1x10^{-8}$.

- **RMSprop (Propagación de Raíz Cuadrática Media):** Es un método de tasa de aprendizaje adaptativo que busca dividir el gradiente por un promedio móvil de su magnitud reciente, busca superar los inconvenientes del algoritmo de Agargrad, que es la acumulación global de caches.

  En las ecuaciones 7.4 y 7.5, se evidencia la fórmula de RMSProp que permite realizar las actualizaciones.

$$E[g^2]_t = \gamma E[g^2]_{t-1} - (1-\gamma) g_t^2 \tag{7.4}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \varepsilon}} g_t \tag{7.5}$$

  Dónde: γ es un valor de 0.9, η es la tasa de aprendizaje, que tiene un valor de 0.001 y $E[g^2]_t$ es el promedio decreciente de los gradientes cuadrados pasados en el paso de tiempo t.

- **Adam (Estimación Adaptativa de Momentos):** Es básicamente una mejora de los optimizadores Adagrad y RMSprop, es un método de aprendizaje adaptativo para cada parámetro, tiene la capacidad de almacenar el promedio decreciente de los gradiente pasados y el promedio decreciente de los gradientes cuadrados anteriores. Para calcular el promedio de los gradientes pasados se utiliza la ecuación 7.6 y para calcular los gradientes cuadrados pasados se utiliza la ecuación 7.7.

$$m_t = \beta_1 m_{t-1} - (1-\beta_1) gt \tag{7.6}$$

$$v_t = \beta_2 v_{t-1} - (1-\beta_2) gt \tag{7.7}$$

  Dado que las ecuaciones 7.6 y 7.7 se inicializan con 0, están sesgadas hacia 0 en especial en los primeros tiempo, razón por la cual, la ecuación 7.6 se corrige y se obtiene la ecuación 7.8 y la ecuación 7.7 se corrige de igual manera y se obtiene la ecuación 7.9.

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{7.8}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^v} \tag{7.9}$$

Y finalmente, en la ecuación 7.10, se evidencia la fórmula de Adam que permite realizar las actualizaciones.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon}\hat{m}_t \tag{7.10}$$

Dónde: β_1 es 0.9, β_2 es 0.999, η es la tasa de aprendizaje y ε es el termino suavizado que permite evitar que se realice una división por cero, en este caso su valor es del orden $1e^{-8}$.

7.1.2.2. Pérdidas

Las funciones de pérdida son métodos que permiten evaluar la capacidad que tienen los algoritmos de aprendizaje profundo para modelar los conjuntos de datos, son medidas que determinan que tan bueno es un modelo para predecir cierto tipo de resultados y se relacionan directamente con las predicciones que el modelo realiza. Estas se agrupan en dos categorías que son de clasificación donde se busca determinar la probabilidad respecto a todas las clases y de regresión donde se busca predecir el valor continuo de un conjunto de datos, a continuación, se muestra una de las principales funciones tanto de probabilidad como de regresión [27][46].

- **Error Cuadrático Medio ('Mean Square Error', MSE):** es una función que permite el cálculo de la distancia geométrica al valor objetivo, es básicamente el promedio de las diferencias al cuadrado entre el valor real y el valor que se espera. La función de costos para este caso es la media de todos los errores cuadráticos.

  En la ecuación 7.11, se evidencia su ecuación.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{7.11}$$

- **Entropía Cruzada Binaria ('Binary Cross Entropy'):** Es un tipo de pérdida por clasificación, con ella se busca determinar la distancia que existe entre las funciones de probabilidad. La pérdida de esta entropía tiende a disminuir en la medida que la probabilidad converge, puede determinar el funcionamiento de un modelo, donde se predicen valores entre 0 y 1.

  En la ecuación 7.12, se evidencia la ecuación que permite determinar la clasificación binaria.

$$L = -\frac{1}{m}\sum_{i=1}^{m}(y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)) \tag{7.12}$$

  En la ecuación 7.13, se evidencia la ecuación que permite determinar la clasificación si hay más de dos clases.

$$L = -\frac{1}{m}\sum_{i=1}^{m}(y_i \cdot \log(\hat{y}_i)) \tag{7.13}$$

  En ambas ecuaciones primero se realiza la división por el número de muestras, luego se realiza la sumatoria de todas las secuencias de cada lote y finalmente se realiza el producto entre la etiqueta real y el logaritmo de la etiqueta que se desea predecir.

7.1.2.3. Funciones de activación

Las funciones de activación son ecuaciones matemáticas que permiten determinar la salida de un modelo de red neuronal, tienen efectos significativos tanto en la capacidad como en la velocidad de convergencia de las redes neuronales, deben ser eficientes y acortar los tiempos de cálculos, de igual manera tienen la función de normalizar las salidas de cualquier entrada en valores de -1 a 1 y 0 a 1. Las funciones de activación no deben ser lineales, deber tener una continuidad diferenciable, deben contar con un rango, deben ser monótonas y deben aproximar la identidad cerca del origen [32][54].

- **Sigmoide:** Son funciones reales diferenciables que son acotadas que se definen para todos los valores de entrada y poseen una derivada no negativa en cada punto, es una función logística y su salida se encuentra entre los rangos de 0 y 1.

  En la ecuación 7.14 y en la figura 7.3a, se evidencia la función de activación sigmoidea o logística y su respuesta.

$$\phi(s_k) = \frac{1}{1 - e^{s_k}} = \frac{e^{s_k}}{1 + e^{s_k}} \tag{7.14}$$

- **Tanh – Tangente hiperbólica:** Es una función continua que genera una salida para cada valor de x y maneja rangos de -1 a 1.

  En la ecuación 7.15 y en la figura 7.3b, se evidencia la función de activación tangente hiperbólica y su respuesta.

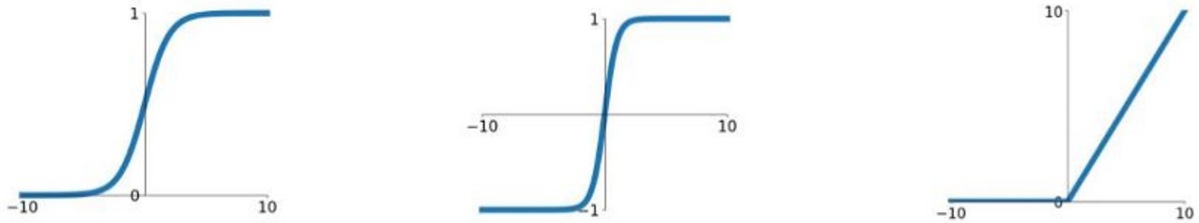
$$\tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})} \tag{7.15}$$

- **ReLU – Unidad Lineal Rectificada:** Es la función de activación más utilizada, al ser una función lineal por partes genera distintos valores de acuerdo a la entrada, en primera medida si el valor es 0 es porque hay una entrada negativa y para los valores positivos de x, se regresan valores positivos.

  En la ecuación 7.16 y en la figura 7.3c, se evidencia la función de activación ReLU y su respuesta.

$$ReLU(x) = max(0, x) \tag{7.16}$$

A continuación, se evidencian las gráficas de las funciones de activación.



(a) Sigmoide (b) Tanh (c) ReLU

Figura 7.3: Funciones de activación [32]

7.1.3. Tipos de Redes Neuronales

Las redes neuronales pueden ser elaboradas de distintas maneras [58]:

1. **Red neuronal prealimentada (Feedforward neural network):** En este tipo de redes neuronales todas las señales van en una única dirección, es decir, desde la entrada hasta la salida, suelen ser utilizadas mayormente para la detección de patrones en específico para la detección de imágenes, se denominan a este tipo de redes neuronales como convolucionales, en la sección 7.1.4, se explica de manera más detallada el funcionamiento de este tipo de red.

2. **Red neuronal realimentada (Feedback neural network):** En este tipo de redes neuronales las señales tienen la capacidad de viajar en varias direcciones y de igual forma pueden generar bucles, este tipo de red es mucho más poderoso y complejo de las de tipo feedforward, se denominan a este tipo de redes neuronales como recurrentes, en la sección 7.1.5, se explica de manera más detallada el funcionamiento de este tipo de red.

3. **Red neuronal autonormalizada (Self-Normalizing Neural Networks):** Son redes que logran su convergencia a un valor cercano a cero en la media y una varianza unitaria incluso en sistemas donde las perturbaciones y el ruido estén presentes. Utilizan funciones de activación SELU (Unidades Lineales Exponenciales Escaladas)[35].

7.1.4. Redes Neuronales Convolucionales

Las redes neuronales convolucionales, son utilizadas para el procesamiento de imágenes, donde su objetivo es aprender de relaciones de entrada-salida, donde en la entrada ingresan imágenes, se basan en la operación de convolución y cumplen funciones como detección de objetos, clasificación de escenarios o de imágenes en general [2]. Para realizar este proceso cada imagen se filtra a través de un kernel, del cual se extraen características representativas de cada una de las imágenes con el fin de realizar una clasificación, todo esto se logra gracias al procesamiento de datos que se realiza a través de una topología de cuadricula, donde cada imagen es una matriz y cada uno de los valores hace referencia a un pixel determinado [40], como se evidencia en la figura 7.4.

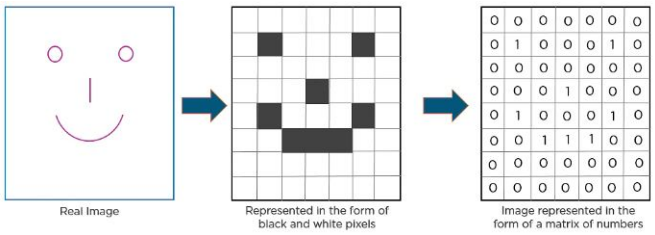


Figura 7.4: Filtrado de imágenes [5]

Las redes neuronales convolucionales cuentan con tipos de capas ocultas:

1. **Capa de convolución:** Se toma una imagen con determinados pixeles y se realiza la convolución con un kernel determinado de 3x3, la idea es realizar la convolución con los grupos de pixeles más cercanos de manera matemática, es decir, a través del producto escalar, si la imagen no posee color solo es necesario emplear un kernel, pero si la imagen es de tipo RGB, debe contar con 3 tipos de kernel, para cada uno de los colores.

2. **Capa de activación:** Para este tipo de redes se emplea la activación de tipo rectificador (ReLU), ya que cuenta con una tasa de aprendizaje adecuada, de igual forma al realizar la operación por elementos puede establecer a los pixeles negativos como 0, es decir, tiene la capacidad de introducir la no linealidad a la red.

3. **Capa de agrupación o reducción:** Permite realizar una reducción del mapa de características a través de una operación de muestreo descendente.

4. **Capa de aplanamiento:** Permite convertir las matrices bidimensionales obtenidas en las operaciones anteriores en mapas de un único vector lineal.

5. **Capa totalmente conectada:** Se conecta al final de toda la red y permite eliminar información espacial irrelevante.

En la figura 7.5, se evidencia toda la estructura básica de una red neuronal convolucional mencionada anteriormente.

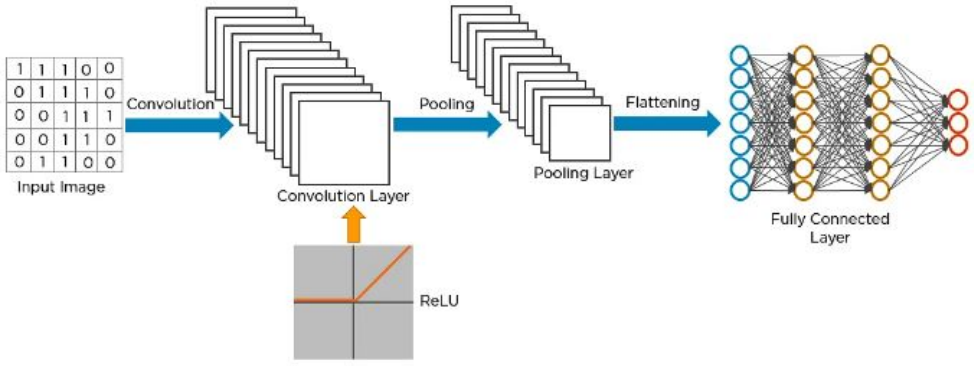


Figura 7.5: Estructura Red Neuronal Convolucional [5]

7.1.5. Redes Neuronales Recurrentes

Las redes neuronales recurrentes tienen la capacidad de procesar datos de manera secuencial logrando que la red tenga en su interior una memoria artificial, tiene la capacidad de predecir datos futuros a partir de datos pasados, este tipo de red tiene la capacidad de guardar su salida en una capa particular para volver a ingresar ese valor y predecir la salida de la capa, en este tipo de red las conexiones forman ciclos, tienen alta potencia dado su estado oculto que posee una dinámica no lineal lo cual le facilita recordar y procesar información [47].

En este caso las redes se encuentran retroalimentadas, lo cual facilita hacer secuencial a lo largo del tiempo. En la figura 7.6 se evidencia la estructura base de una red neuronal recurrente, donde "x" hace referencia a la capa de entrada que toma la entrada inicial y realiza un procesamiento, "h" a la capa oculta que maneja todo lo relacionado con las capas ocultas, las funciones de activación y todos los pesos y "y" la capa de salida, en este caso para determinar los valores de salida es necesario tomar en cuenta el tiempo pasado, presente y futuro, como se evidencia en la figura 7.6.

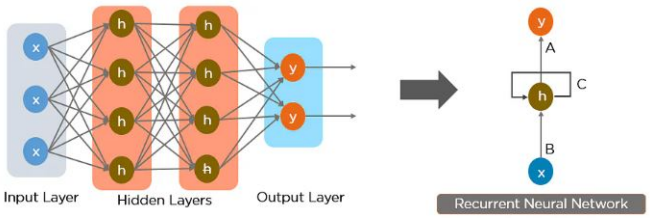


Figura 7.6: Estructura Red Neuronal Recurrente [6]

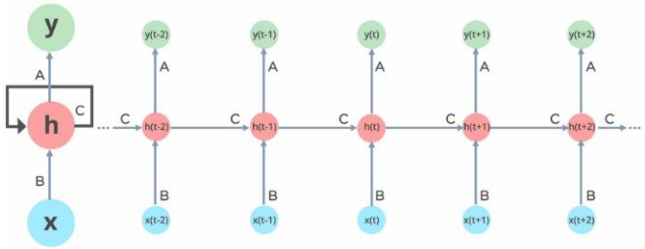


Figura 7.7: Red Neuronal Recurrente totalmente conectada [6]

Este tipo de red se representa a partir de la ecuación 7.17, donde h(t) hace referencia al nuevo estado de salida, f_c es la función del parámetro que atraviesa toda la red, h(t-1) es el valor pasado y x(t) es el vector de entrada para un tiempo t[6].

$$h(t) = f_c(h(t-1), x(t)) \tag{7.17}$$

Las redes recurrentes tienen varios tipos, como el de una entrada y una salida, una entrada y varias salidas, varias entradas y una salida y varias entradas y varias salidas. Este tipo de redes suele tener 2 inconvenientes muy marcados, que son el problema de gradiente de fuga, ya que mientras el gradiente se va reduciendo los parámetros que lo conforman también lo hacen y se vuelven poco relevantes y el otro inconveniente es el problema del gradiente explosivo, ya que la pendiente en lugar de crecer de manera exponencial comienza a decaer, esto suele ocurrir debido al exceso de entrenamiento o la mala precisión, para este último caso, se genera un nuevo tipo de red denominado, LSTM, que es un tipo de red de memoria a corto plazo [2][6].

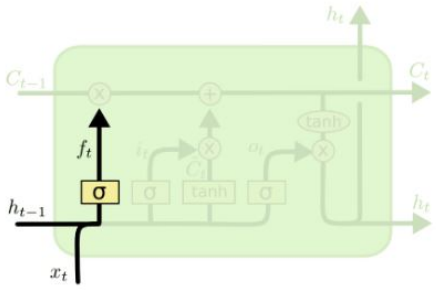
7.1.5.1. Redes Neuronales Recurrentes tipo LSTM

Las redes neuronales recurrentes de tipo LSTM, tienen la capacidad de manejar dependencias a largo plazo, es decir, tienen la capacidad de recordar información a largo plazo, son redes secuenciales avanzadas, este tipo de redes consta de tres partes principales y una celda para los estados [2][36][45][55][61]:

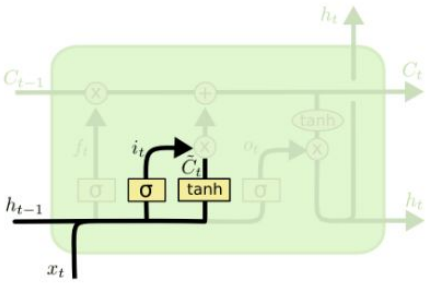
1. **Forget Gate:** Determina que información es o no es relevante, se evidencia su funcionamiento en la figura 7.8a y se basa en la ecuación 7.18:

$$f_t = \sigma(W_f \cdot [H_{t-1}, x_t] + b_f) \tag{7.18}$$

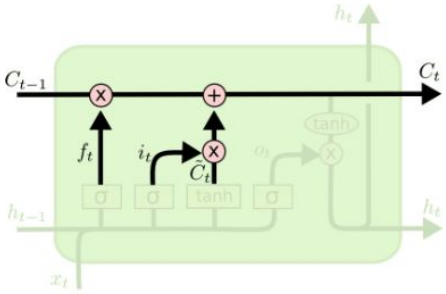
Dónde: x_t ingresa a la marca de tiempo actual, b_f es el valor del bias, H_{t-1} es el estado oculto que evidencia el peso anterior y W_f es una matriz de peso que se asocia al estado oculto.



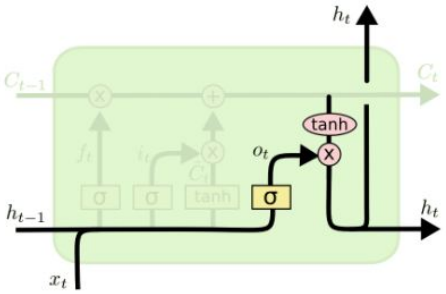
(a) Compuerta FORGET [45]



(b) Compuerta INPUT [45]



(c) Celda de Estado [45]



(d) Compuerta OUTPUT [45]

Figura 7.8: Composición de una Red Neuronal Recurrente de tipo LSTM

A esta ecuación se le aplica una función sigmoidea que permite que el valor sea 0 o 1.

Si se quiere olvidar todo, se emplea la ecuación 7.19, si se desea no olvidar nada se emplea la ecuación 7.20.

$$C_{t-1} * f_t = 0 \text{ si } f_t = 0 \tag{7.19}$$

$$C_{t-1} * f_t = 0 \text{ si } f_t = 1 \tag{7.20}$$

Se busca determinar qué información no es necesario para el estado de la celda.

2. **Input Gate:** Aprende información nueva y calcula el nivel de importancia de la nueva información que se transporta a la entrada, se evidencia su funcionamiento en la figura 7.8b y se basa en la ecuación 7.21:

$$i_t = \sigma(W_i \cdot [H_{t-1}, x_t] + b_i) \tag{7.21}$$

Dónde: x_t ingresa a la marca de tiempo actual, b_f es el valor del bias, H_{t-1} es el estado oculto que evidencia el peso anterior y W_f es una matriz de peso que se asocia al estado oculto. En este caso una vez se deciden los valores a actualizar, se toma una capa tanh para crear nuevo vectores candidatos, como se evidencia en la ecuación 7.22.

$$\tilde{C} = \tanh(W_C[h_{t-1}, x_t] + b_C) \tag{7.22}$$

Para esto se opera lo obtenido en la ecuación 7.18 y luego se le agrega la multiplicación de las ecuaciones 7.21 y 7.22, para que esos nuevos valores sean los candidatos, para poder actualizar el valor del estado, como se evidencia en la ecuación 7.23 y se puede apreciar su funcionamiento en la figura 7.8c.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{7.23}$$

3. **Output Gate:** Permite el paso de información altamente relevante a la salida, se evidencia su funcionamiento en la figura 7.8d y para este caso se emplea la ecuación 7.25.

$$o_t = \sigma(W_o \cdot [H_{t-1}, x_t] + b_o) \tag{7.24}$$

Donde: x_t ingresa a la marca de tiempo actual, b_o es el valor del bias, H_{t-1} es el estado oculto que evidencia el peso anterior y W_o es una matriz de peso que se asocia al estado oculto. Una vez se tienen estos valores se pasa la celda a través de una tanh, como se evidencia en la ecuación 7.25, para colocar valores entre -1 y 1.

$$h_t = o_t * \tanh(C_t) \tag{7.25}$$

Gracias a sus 3 compuertas es posible tener una red neuronal recurrente mejorada, ya que es posible eliminar información irrelevante y tener presente información específica, todo esto gracias a su memoria tanto de corto como de largo plazo.

En la figura 7.9, se evidencia el diagrama de la red.

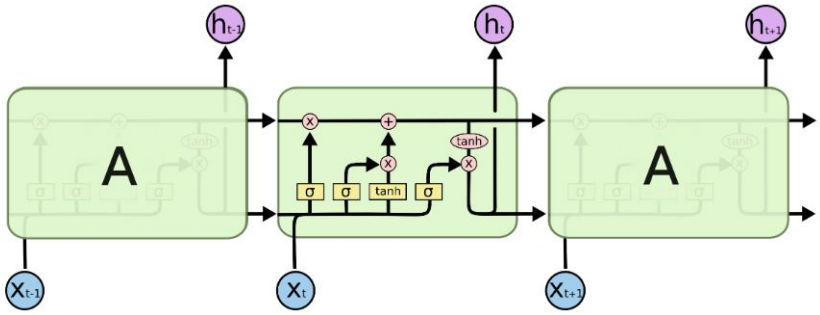


Figura 7.9: Arquitectura Red Neuronal Recurrente tipo LSTM [45]

Capítulo 8

Metodologia

8.1. Etapas y metodología

El proyecto implementado se basa en una investigación y estudios experimentales y empíricos. Con el propósito de ejecutar a cabalidad los objetivos propuestos en la sección 6 para la implementación del proyecto, se ejecuta la siguiente metodología de trabajo presentada en la figura 8.1. que consta de 5 etapas.

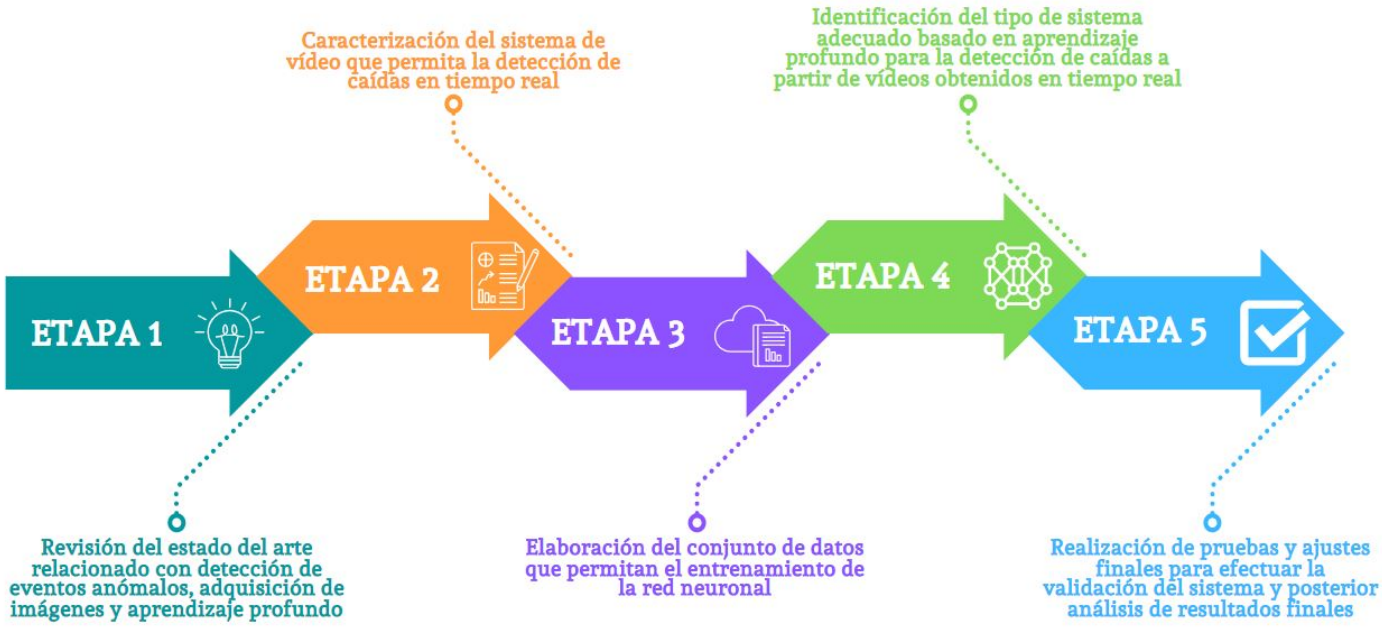


Figura 8.1: Metodología del proyecto

8.1.1. Primer etapa.

Revisión del estado del arte relacionado con caídas de personas mayores y proyectos implementados con diferentes tipos de arquitecturas de redes neuronales, de igual forma se realiza un estudio detallado de todas las temáticas referentes a las redes neuronales, como lo son su estructura, arquitecturas y tipos de redes neuronales, todo esto a partir de los repositorios institucionales como IEEE Explorer, SpringerLink y de igual manera en Google Académico.

8.1.2. Segunda etapa.

Revisión detallada de las especificaciones y características de cómputo y de la cámara, al igual que las especificaciones de software, como lo son las librerías y el lenguaje de programación que se va a emplear para la ejecución del proyecto.

8.1.3. Tercera etapa.

Toma, detección y extracción de puntos de referencia de las acciones a detectar a partir de las posturas del cuerpo humano y puntos específicos del cuerpo, con ayuda de MediaPipe, se realiza la recolección de todos los datos y un posterior procesamiento de cada uno de ellos, ya que estos serán los datos de entrada de la red neuronal que se va a implementar.

8.1.4. Cuarta etapa.

Basado en la revisión del estado del arte, se implementa el algoritmo que obtuvo mejores resultados para la detección de caídas en tiempo real, para ello se realiza el diseño y posterior implementación del algoritmo basado en redes neuronales.

8.1.5. Quinta etapa.

Se evalúa el sistema en su totalidad y se verifica el funcionamiento del algoritmo implementado, con el fin de evidenciar si se presentaron algún tipo de errores o inconvenientes en la implementación realizada.

8.2. Cronograma

Para la realización de toda la metodología, se realiza el cronograma de la tabla 8.1 donde se realizan actividades para cada uno de los objetivos específicos de la sección 6.2, con el fin de cumplir a cabalidad el objetivo general de la sección 6.1.

Objetivos Específicos	Actividades	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Mes 6	Mes 7	Mes 8
1	1.1. Revisión de temáticas relacionadas con caídas, factores de riesgo y consecuencias	X	X						
	1.2. Revisión de proyectos implementados con RN para la detección de caídas	X	X						
2	2.1. Revisión de específicaciones del hardware (Computador y cámara)			X					
	2.2. Revisión de específicaciones del software			X					
3	3.1. Toma, detección y extracción de puntos de referencia de las acciones a detectar				X				
	3.2. Recolección de los datos				X				
	3.3. Procesamiento de los datos				X				
4	4.1. Desarrollar la aplicación con un algoritmo basado en redes neuronales					X	X	X	
	4.2. Pruebas de funcionamiento y detección de errores					X	X	X	
5	5.1. Ejecutar la revisión final y funcionamiento adecuado del algoritmo implementado								X

Tabla 8.1: Cronograma del proyecto

Capítulo 9

Desarrollo Conceptual

Para la ejecución de la metodología expuesta en la sección 8.1, se proceden a ejecutar las actividades propuestas en el cronograma de la sección 8.2.

9.1. Actividades para objetivo específico 1

9.1.1. Actividad 1.1 - Caídas

El objetivo del proyecto es realizar la detección de caídas en personas vulnerables, en especial, en personas mayores, para ello es necesario tener presente que puede provocar este tipo de sucesos, al igual que es necesario evidenciar que tipo de factores de riesgo y consecuencias se pueden presentar al ocurrir una caída, es por ello que en la sección 3.0.1 del capítulo 3, se puede evidenciar este estudio.

9.1.2. Actividad 1.2 - Proyectos implementados para la detección de caídas

Por otro lado, es necesario hacer una revisión del estado del arte de proyectos implementados que empleen diferentes tipos de arquitecturas de redes neuronales tanto convolucionales como recurrentes y revisar el funcionamiento e implementación de cada una de ellas, tanto de manera matemática como en código, es por ello que en la sección 3.0.2 del capítulo 3, se puede evidenciar los proyectos implementados con redes neuronales y en el capítulo 7, se puede evidenciar el estudio detallado de todo lo relacionado con redes neuronales.

9.2. Actividades para objetivo específico 2

9.2.1. Actividad 2.1 - Especificaciones de hardware

Todas las características del hardware tanto para la parte del sistema de cómputo como para la cámara empleados para el desarrollo del proyecto se evidencian en la tabla 9.1 y en la tabla 9.2 respectivamente.

- Especificaciones del sistema de cómputo.

Recurso		Especificación
Procesador	CPU	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz
Memoria	RAM	16 GB
Tarjeta gráfica	GPU	Intel iRIS x^e Graphics

Tabla 9.1: Específicaciones de hardware (Computador)

- Especificaciones de la cámara.

Recurso		Especificación
Resolusión máxima		720p / 30fps
Cámara mega píxel		0.9
Tipo de enfoque		Foco fijo
Campo visual diagonal	dFoV	55°

Tabla 9.2: Específicaciones de hardware (Cámara)

9.2.2. Actividad 2.2 - Especificaciones de software

Todas las características del software empleado para el desarrollo del proyecto se evidencian en la tabla 9.3.

Recurso	Especificación	Versión
UBUNTU	Sistema Operativo	20.04
Python	Lenguaje de programación	3.9.12
OpenCV	Libería de visión artificial	4.6.0
Numpy	Librería para cálculo numérico	1.21.5
Matplotlib	Librería de visualización de datos	3.5.1
MediaPipe	Solución de aprendizaje máquina para detección en 3D	0.8.10.1
Scikit-Learn	Librería de aprendizaje automático y modelado estadístico	1.0.2
Seaborn	Librería para visualización de datos estadísticos	0.11.2
TensorFlow	Libería de aprendizaje automático para redes neuronales	2.9.1
Keras	Librería para construcción de arquitecturas de redes neuronales	2.9.0

Tabla 9.3: Especifícaciones de software

9.3. Actividades para objetivo específico 3

Con el fin de realizar la toma de datos para realizar el entrenamiento de la red neuronal, es necesario emplear un detector de posturas que no solo permita determinar la posición sino también la ubicación de las personas. Para ello se procede a realizar la comparación de dos de los detectores de postura más utilizados para proyectos de visión por computadora, con ayuda

del artículo denominado “Comparing the Quality of Human Pose Estimation with BlazePose or OpenPose” [42] y el artículo “Detection of human body landmarks - MediaPipe and OpenPose comparison” [49].

	BlazePose	OpenPose
Velocidad de procesamiento de vídeo	140 fps	22 fps
Umbral de correlación	0.8	0.8
Mayor tiempo de ejecución	X	✓
Procesamiento de vídeos de entrada en tiempo real	✓	X
Mejor respuesta a iluminación desfavorable	✓	X
Mejor presición al detectar puntos de referencia	✓	✓

Tabla 9.4: Comparación estimadores de postura

Con base en la información anterior, se evidencia que MediaPipe, cuenta con mejores resultados en lo relacionado a velocidad de procesamiento de vídeo, menor tiempo de ejecución, procesamiento de vídeos en tiempo real, mejor respuesta a mala iluminación y sobre todo buena precisión al realizar detección de los puntos de referencia, con base en ello a continuación, se evidencian las principales características de esta herramienta.

9.3.1. MediaPipe

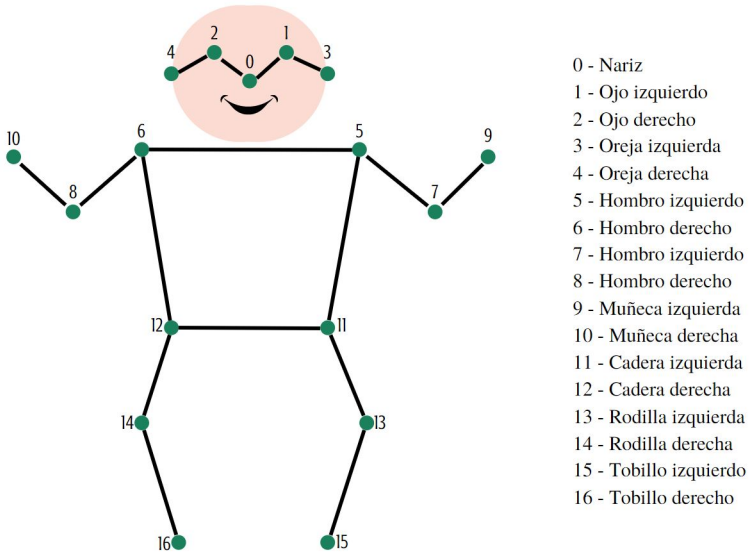
MediaPipe es una librería de aprendizaje profundo y visión computacional que se basa en la detección de distintas partes del cuerpo humano, esta herramienta cuenta con un detector que tiene la capacidad de identificar las áreas de interés en los vídeos en tiempo real y un rastreador que permiten determinar los distintos puntos característicos del cuerpo humano. Las soluciones de aprendizaje máquina de MediaPipe van desde la detección facial, la detección del iris, detección de manos, detección de postura, solución holística, detección de objetos, entre otros [26].

Para el caso particular del proyecto es necesario utilizar la detección de la postura, que incluye detección facial y la detección de manos.

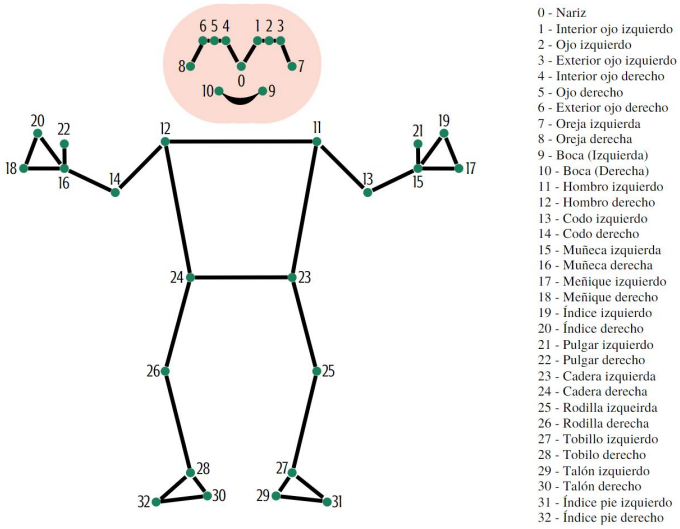
9.3.1.1. MediaPipe Pose

Esta solución para el seguimiento que postura cuenta con una alta fidelidad y se realiza con ayuda del modelo ligero BlazePose, la cual toma en consideración 33 puntos distintos de referencia en segunda dimensión a partir de un único fotograma, lo que permite la localización precisa de los puntos clave [39].

Topología: El estándar actual para la postura del cuerpo humano es COCO (Common Objects in Context), que consta únicamente de 17 puntos de referencia, como se evidencia en la figura 9.1a, que se obtienen a partir de la localización de los puntos de claves de las personas en condiciones desafiantes y no controladas [38], a partir de ello se obtienen puntos de referencia para el torso, los brazos, la piernas y la cara, un problema de COCO, es que no cuenta con la capacidad de determinar la ubicación real de los tobillos y las muñecas, que a largo plazo resulta vital para realizar otro tipo de modelos de estimación de posturas.



(a) Puntos clave estándar COCO



(b) Puntos clave modelo BlazePose

Figura 9.1: Puntos clave modelos COCO y BlazePose.

Pero gracias la creación de BlazePose, fue posible implementar una nueva topología con los 33 puntos clave del cuerpo humano, como se evidencia en la figura 9.1b, esto se logra gracias a la unión de las topologías, COCO, BlazeFace que permite asignar mayor cantidad de puntos al rostro y BlazePalm, que permite determinar los puntos de los tobillos y las muñecas.

Seguimiento de poses: Para realizar esta detección es necesario emplear una tubería de aprendizaje automático que permita el seguimiento y estimación de posturas, la idea es que a través del detector sea posible como primera medida determinar la región de interés (ROI) de la postura dentro de un marco, para que posterior a ello sea posible determinar la ubicación de los 33 puntos claves, cabe recalcar que esto se hace únicamente para el primer frame del vídeo, posterior a ello se reasigna la ROI de los puntos claves del fotograma anterior, como se evidencia en la figura 9.2.

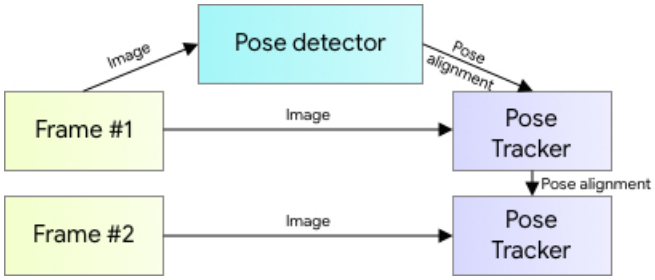


Figura 9.2: Tubería para estimación de la postura [4]

Detección de posturas a través de BlazePose: Para que esta detección se realice de manera rápida, se toma como referencia la cara y el tronco como la señal más fuerte de detección para la red neuronal, con el fin de realizar el modelo de seguimiento para la estimación de la posición se deben tener en cuenta los puntos de referencia que consisten en los 3 grados de libertad (x, y, z) que hacen referencia a la ubicación, dónde x y y hacen referencia a las coordenadas normalizadas para el ancho y el alto de la imagen, z hace referencia a la profundidad del punto de referencia con la profundidad en el punto medio de las caderas como origen y la visibilidad que indica la probabilidad de que el punto de referencia sea o no visible en la imagen, al igual que estos puntos de referencia también se deben tomar como referencia los puntos tanto de la cara como del tronco. Este modelo realiza la predicción a partir de un enfoque de regresión supervisado por una predicción que combina el mapa de calor de cada uno de los puntos clave. Para el entrenamiento se emplea el mapa del calor y la pérdida compensada y luego se elimina el mapa de calor y se realiza el entrenamiento del codificador de regresión [4], cabe recalcar que al momento de realizar la toma del vídeo del tiempo real, el frame toma un tamaño inicial de 256x256 píxeles para realizar el entrenamiento de la red como se muestra en la arquitectura de la figura 9.4.

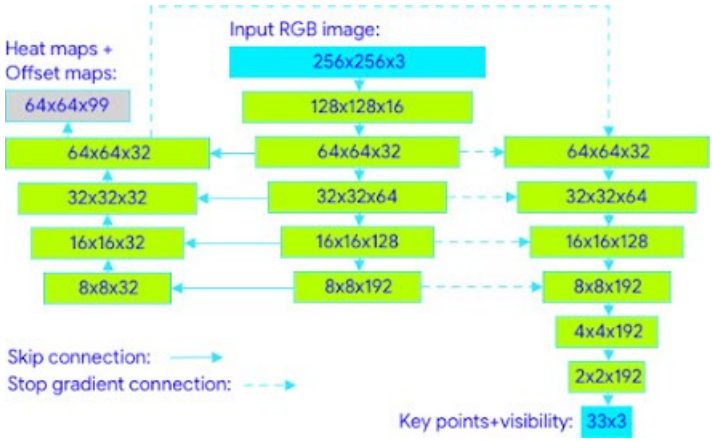


Figura 9.3: Arquitectura para detección de posturas [4]

9.3.2. Actividad 3.1 - Toma, detección y extracción de puntos de referencia

Para la toma, detección y extracción de los puntos de referencia obtenidos gracias al detector de posturas, se sigue en flujo de la figura 9.4.

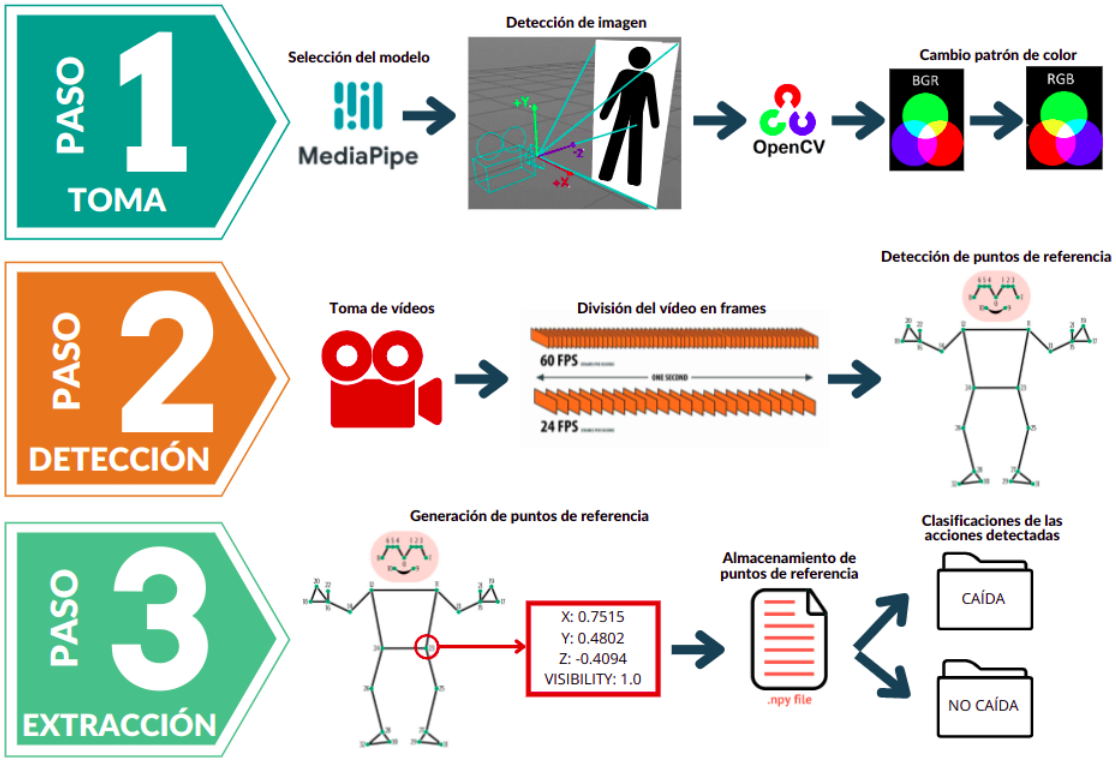


Figura 9.4: Arquitectura para detección de posturas

A continuación, se explica de manera detallada cada uno de los pasos del flujo anterior.

9.3.2.1. Toma de puntos de referencia.

Para realizar la toma de los puntos de referencia a través de vídeos en tiempo real, es necesario determinar en primera medida el modelo que se desea emplear, en este caso la solución holística que ofrece Mediapipe (mp.solutions.holistic) y la utilidad de dibujo que se requiere para poder tomar los puntos de referencia que de igual forma se generan de la solución de MediaPipe (mp.solutions.drawing_utils) [50]. El objetivo es realizar la detección de imagen con ayuda de OpenCV, convirtiendo los colores de BGR (Blue, Green, Red) a RGB (Red, Green, Blue), con el fin de poder hacer que la imagen sea procesada de manera adecuada y se logre la detección, en la figura 9.5, se evidencia el cambio de color que se debe generar con cada frame del vídeo para llegar a la detección (Ver anexo 11).

Figura 9.5: Cambio de color de imagen de BGR a RGB [12]

Como se evidencia en la sección 9.3.1, el modelo holístico de MediaPipe, permite realizar conexiones para el cuerpo, la cara, las manos y los pies, como se muestra en la figura 9.6, en este caso las conexiones se muestran punto por punto de acuerdo a lo que la cámara logra adquirir, cuando se realiza la primera prueba de toma de vídeo.

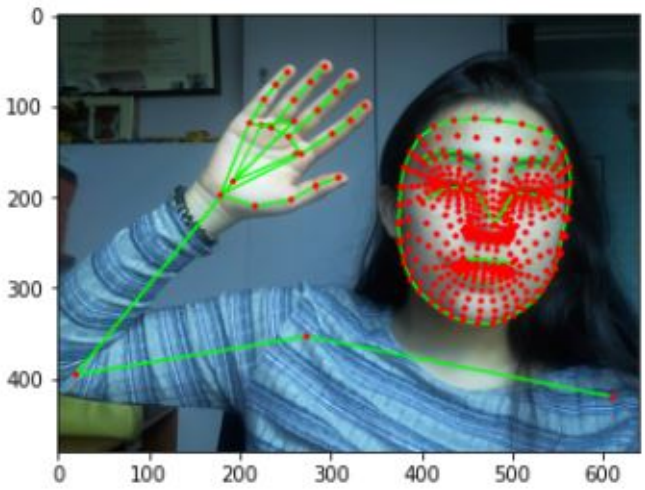


Figura 9.6: Conexiones del cuerpo.

A partir de dichas conexiones que se toman por cada frame de vídeo, se obtienen las conexiones punto a punto, como se muestra en la figura 9.7.

```
frozenset({(<PoseLandmark.NOSE: 0>, <PoseLandmark.LEFT_EYE_INNER: 1>),
           (<PoseLandmark.NOSE: 0>, <PoseLandmark.RIGHT_EYE_INNER: 4>),
           (<PoseLandmark.LEFT_EYE_INNER: 1>, <PoseLandmark.LEFT_EYE: 2>),
           (<PoseLandmark.LEFT_EYE: 2>, <PoseLandmark.LEFT_EYE_OUTER: 3>),
           (<PoseLandmark.LEFT_EYE_OUTER: 3>, <PoseLandmark.LEFT_EAR: 7>),
           (<PoseLandmark.RIGHT_EYE_INNER: 4>, <PoseLandmark.RIGHT_EYE: 5>),
           (<PoseLandmark.RIGHT_EYE: 5>, <PoseLandmark.RIGHT_EYE_OUTER: 6>),
           (<PoseLandmark.RIGHT_EYE_OUTER: 6>, <PoseLandmark.RIGHT_EAR: 8>),
           (<PoseLandmark.MOUTH_RIGHT: 10>, <PoseLandmark.MOUTH_LEFT: 9>),
           (<PoseLandmark.LEFT_SHOULDER: 11>, <PoseLandmark.LEFT_ELBOW: 13>),
           (<PoseLandmark.LEFT_SHOULDER: 11>, <PoseLandmark.LEFT_HIP: 23>),
```

Figura 9.7: Conexiones del cuerpo.

9.3.2.2. Detección de puntos de referencia.

Una vez se tienen presentes las conexiones generadas en la sección 9.3.2.1, es necesario generar una función que permita dibujar las marcas y las 33 conexiones que conforman la estructura del cuerpo, donde 11 puntos son de la cara, 4 puntos para cada una de las manos, 4 puntos para cada pie y el restante de los punto para el cuerpo, como se puede evidenciar en la tabla 9.10 y se complementa junto con la figura 9.1b.

Conexiones del cuerpo					
Cara	Cuerpo	Manos		Pies	
0,1,2,3,4,5, 6,7,8,9,10	11,12,13,14, 23,24,25,26	Mano Derecha	16,18,20,22	Pie Derecho	27,29,31
		Mano Izquierda	15,17,19,21	Pire Izquierdo	28,30,32

Tabla 9.5: Puntos de referencia del cuerpo

Una vez detectados todos los puntos, se procede a generar círculos en las articulaciones y colores entre las conexiones, esto se realiza con la ayuda de un vídeo tomado en tiempo real, en el cuál es posible emplear el modelo holístico de MediaPipe, en el cual mientras se va realizando la captura de vídeo se van adquiriendo todos los puntos de referencia, en la figura 9.8, se puede evidenciar un frame del vídeo dónde en color rojo se evidencian los puntos de todas las articulaciones y en blanco las líneas que permiten las conexiones entre cada punto de referencia.

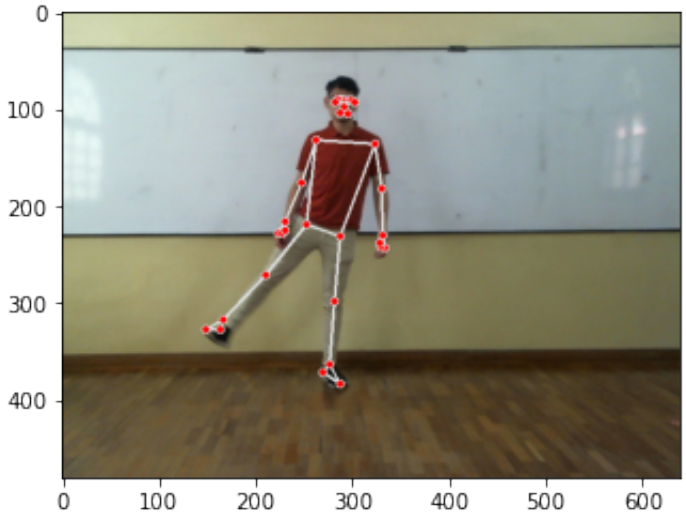


Figura 9.8: Detección de puntos de referencia en frame de vídeo en tiempo real.

9.3.2.3. Extracción de puntos de referencia.

Con la toma de los datos obtenidos en la sección 11, es necesario que se generen los puntos de referencia con sus 4 coordenadas determinadas (x, y, z, visibilidad), como se evidencia en la figura 9.9 (Ver anexo 11).

- **Puntos de referencia del cuerpo:**

  En la variable donde se almacenan los valores, se toman 4 puntos de referencia (x, y, z, visibilidad) por marca, como se muestra en la figura 9.9.

Con base en lo anterior, es necesario concatenar todos los valores con el objetivo de realizar una extracción de datos para generar un arreglo, que es el que nos va a servir de entrada para la red neuronal que se va a entrenar, para ello se genera una función que permite extraer los puntos de los 4 tipos de puntos (x, y, z, visibilidad) de referencia que se toman, en este caso la idea es que para cada uno de los frames del vídeo se realice un almacenamiento de absolutamente todos los puntos y estos logren ser guardados en un archivo de tipo Numpy (.npy) [50].

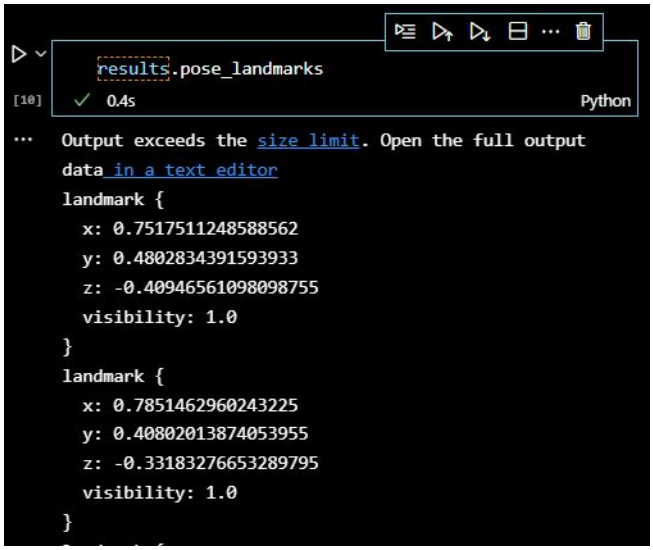


Figura 9.9: Puntos de referencia adquiridos (cuerpo).

9.3.3. Actividad 3.2 - Recolección de los datos

Para realizar la correcta recolección de todos los datos primero es necesario crear dos carpetas, cada una de ellas va a contener una de las acciones que se van a detectar, es decir, caída y no caída, la idea es tomar un total de 50 vídeos cada uno de 50 frames, para cada acción y cada uno de esos vídeos va a tener asignada una carpeta y dentro de cada carpeta se van a guardar los puntos de referencia adquiridos en cada frame (Ver anexo 11).

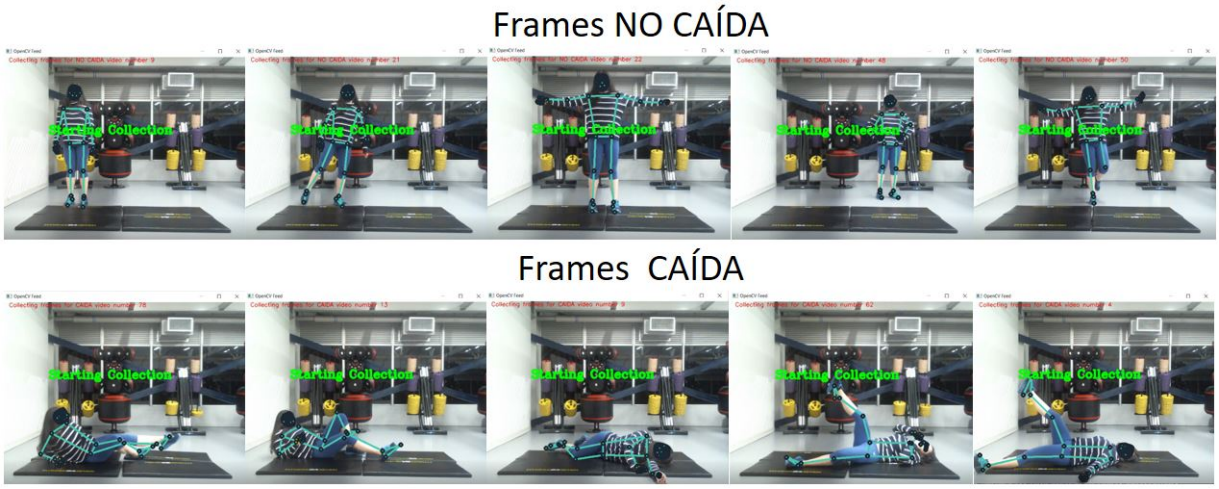


Figura 9.10: Frames obtenidos.

Para realizar el entrenamiento de la red neuronal, es necesario almacenar datos, en este caso, se almacenan datos en arreglos que se guardan en archivos de tipo *numpy*, que permiten realizar el entrenamiento de esta y posterior evaluación de su funcionamiento, para ello se realiza la toma de los vídeos en ráfaga, que almacenan los puntos de referencia en las carpetas respectivas, de acuerdo a cada una de las acciones generadas. Se realiza la captura de vídeo, mientras que de cada uno de los frames del vídeo se toman los datos de las posturas y después se almacenan de manera

consecutiva en cada una de las carpetas, primero se realiza la toma de los 50 vídeos de caídas y posterior a ello se realiza la toma de los otros 50 vídeos de no caídas, en la figura 9.10 se evidencian algunos de los frames obtenidos para cada una de las acciones a detectar.

9.3.4. Actividad 3.3 - Procesamiento de los datos

Al tener las carpetas organizadas se determinan las etiquetas de las imágenes, para el caso de la acción de "NO CAÍDA" se toma el valor de 0, mientras que para la acción de "CAÍDA" se toma el valor de 1. Una vez se tienen estos datos, se genera una función que permita crear un arreglo con 132 valores, obtenidos de los puntos de referencia, es decir, para la postura son 33 puntos por 4 dimensiones, con estos datos, se busca que cada acción contenga todos los valores de las secuencias encontradas [50].

Con base en lo anterior se obtiene que el número de secuencias viene dado por (100, 50, 132) como se evidencia en la figura 9.11, donde 100 es la cantidad de vídeos, 50 la cantidad de frames por vídeo y 132 todos los puntos de referencia (Ver anexo 11).

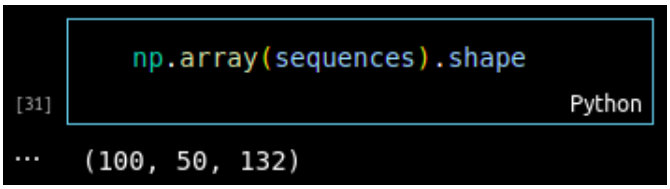
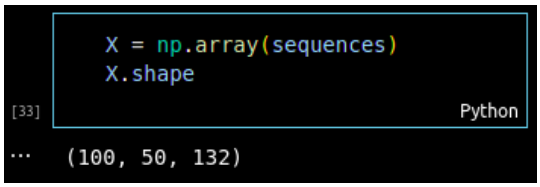
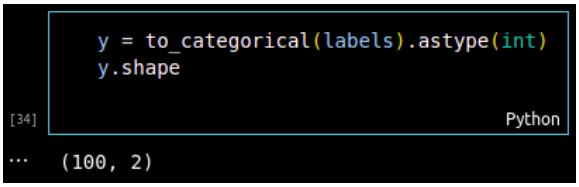


Figura 9.11: Valores obtenidos para las secuencias de vídeo.

Estos datos, deben ser organizados que sea posible tomar una parte para realizar el entrenamiento de la red y la otra parte para hacer la evaluación de esta, es por eso, que se deben sacar los parámetros para "x" y para "y". En el caso de los valores de "x" se toman los datos del números de secuencias de la figura 9.12a, mientras que para los valores de "x", como se muestra en la figura 9.12b se toman los correspondientes a las etiquetas, es decir, 100 vídeos y 2 etiquetas.



(a) Valores para "x"



(b) Valores para "y"

Figura 9.12: Valores para "x" y "y"

Para realizar la división de los datos entre entrenamiento y evaluación, se emplea la librería scikit-learn, se utiliza el modelo de selección train_test_split que consta de 3 parámetros (los valores de x, los valores de y, el porcentaje de datos de evaluación), dónde se buscan obtener 4 parámetros a partir de los 3 anteriores, en este caso son 2 para entrenamiento (X_train, y_train) y 2 para evaluación (X_test, y_test) y se obtiene lo siguiente:

- Para el valor de "x" de entrenamiento de la figura 9.13, se toman 80 de los 100 vídeos adquiridos.

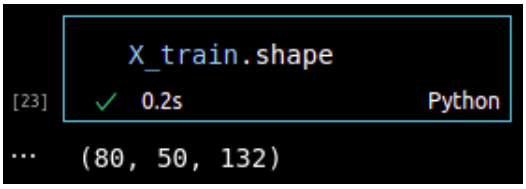


Figura 9.13: Valores de “x” para entrenamiento.

- Para el valor de “y” de entrenamiento de la figura 9.14, se toman 80 de las 100 etiquetas adquiridas.

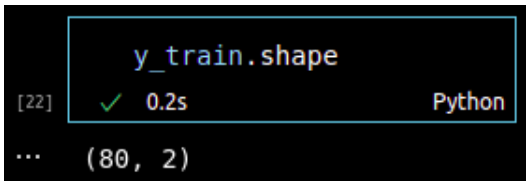


Figura 9.14: Valores de “y” para entrenamiento.

- Para el valor de “x” de prueba de la figura 9.15, se toman 20 de los 100 vídeos adquiridos.

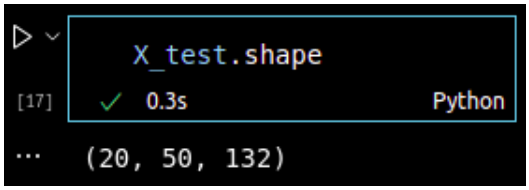


Figura 9.15: Valores de “x” para evaluación.

- Para el valor de “y” de prueba de la figura 9.16, se toman 20 de las 100 etiquetas adquiridas.

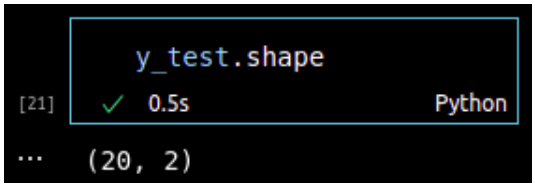


Figura 9.16: Valores de “y” para evaluación.

9.4. Actividades para objetivo específico 4

De acuerdo a lo evidenciado en la sección 3.0.1 del capítulo 3, se observa que los proyectos que mejor respuesta generar para la detección de caídas en tiempo real, son los que fueron elaborados con redes neuronales recurrentes de tipo LSTM.

9.4.1. Actividad 4.1 - Implementación del algoritmo basado en Redes Neuronales

A continuación, se evidencia de manera detallada cada uno de los pasos que se ejecutaron para la implementación, desarrollo y entrenamiento de la red neuronal (Ver anexo 11).

1. **Carga del conjunto de datos:** Como se evidencia en la sección 3, luego de realizar la toma de los datos, clasificarlos, concatenarlos y ubicarlos en las carpetas de cada una de las acciones, es posible tener los parámetros iniciales para realizar el entrenamiento de la red.

2. **Definición del modelo:** Para la implementación de la Red Neuronal se utiliza el método "Sequential", que es una clase secuencial de Keras que tiene la capacidad de agrupar en pila todas las capas de una Red Neuronal, para el caso de la definición del modelo se emplean las tres tipos de capas diferentes, LSTM, Dropout y Dense.

   - **Capa LSTM:** Este parámetro cuenta con las siguientes variables:
     - **Unidades:** Son valores positivos, que manejan la dimensionalidad del espacio de salida que se desee.
     - **input_shape:** Es la dimensión de la característica de entrada de la red, en este caso los valores de los frames del vídeo y los puntos de referencia de todo el cuerpo junto con sus coordenadas.
     - **return_sequences:** Permite devolver el valor de cada paso de tiempo.

     Para el caso del modelo a implementar se manejan las siguientes 4 capas de tipo LSTM, que poseen los siguientes valores:

Capa	Unidades	return_sequences	input_shape
LSTM 1	64	Verdadero	(50, 132)
LSTM 2	128	Verdadero	NA
LSTM 3	64	Verdado	NA
LSTM 4	32	Falso	NA

Tabla 9.6: Parámetros capa LSTM

   - **Capa Dropout:** Este parámetro cuenta con la siguiente variable:
     - **Rate:** Se configura para evitar el sobreajuste y se configura con valores de entre 0 y 1, de acuerdo a lo que el modelo lo requiera.

     Para el caso del modelo a implementar se manejan las siguientes 4 capas de tipo Dropout, que poseen los siguientes valores:

Capa	Rate
Dropout 1	0.2
Dropout 2	0.2
Dropout 3	0.2
Dropout 4	0.2

Tabla 9.7: Parámetros capa Dropout

- **Capa Dense:** Este parámetro cuenta con las siguientes variables:
  - **Unidades:** Son valores positivos, que manejan la dimensionalidad del espacio de salida que se desee.
  - **Activación:** Es el tipo de activación que va a tener el modelo al ser entrenado.

En la siguiente tabla, se muestra la relación de los valores seleccionados para la implementación de la red.

Capa	Unidades	Activación
Dense 1	64	relu
Dense 2	32	relu
Dense 3	32	relu
Dense 4	actions.shape[0]	sigmoide

Tabla 9.8: Parámetros capa Dense

3. **Compilación del modelo:** Para realizar el entrenamiento del modelo se utiliza el método "Compile", este parámetro cuenta con las siguiente variables:

   - **Optimizer:** Es el tipo de optimizador que se va a emplear.
   - **Loss:** Son las funciones de pérdida.
   - **Métrics:** Este parámetro evaluará el modelo a lo largo del entrenamiento y las pruebas.

   En la siguiente tabla, se muestra la relación de los parámetros seleccionados para compilar el modelo:

Parámetro	Tipo
optimizer	Adam
loss	binary_crossentropy
metrics	accuracy

Tabla 9.9: Parámetros de compilación del modelo

4. **Ajuste del modelo:**

   Para realizar el ajuste del modelo se utiliza el método "Fit", este parámetro cuenta con las siguientes variables:

   - **X:** Son los datos de entrada.
   - **Y:** Son los datos de destino
   - **epochs:** Es el número de épocas que se utilizan para entrenar el modelo, este valor genera iteraciones sobre la totalidad de los valores de "x" y "y".
   - **batch_size:** Es el número de muestras por actualización de gradiente.
   - **validation_data:** Son los datos sobre los que se puede evaluar la pérdida y cualquier métrica del modelo al final de cada época.

En la siguiente tabla, se muestra la relación de los parámetros seleccionados para compilar el modelo:

Parámetro	Valor
X	X_train
Y	y_train
epochs	16
batch_size	32
validation_data	(X_test, y_test)

Tabla 9.10: Parámetros de ajuste del modelo

En la imagen 9.18, se evidencia el resumen del algoritmo implementado anteriormente.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 50, 64)            50432

 dropout (Dropout)           (None, 50, 64)            0

 lstm_1 (LSTM)               (None, 50, 128)           98816

 dropout_1 (Dropout)         (None, 50, 128)           0

 lstm_2 (LSTM)               (None, 50, 64)            49408

 dropout_2 (Dropout)         (None, 50, 64)            0

 lstm_3 (LSTM)               (None, 32)                12416

 dropout_3 (Dropout)         (None, 32)                0

 dense (Dense)               (None, 64)                2112

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 32)                1056

 dense_3 (Dense)             (None, 2)                 66

=================================================================
Total params: 216,386
Trainable params: 216,386
Non-trainable params: 0
```

Figura 9.17: Resumen de la red implementada.

9.4.2. Actividad 4.2 - Pruebas de funcionamiento y detección de errores

Con el objetivo de realizar las pruebas de funcionamiento y detección de errores, se implementa el flujo de la figura 9.18, el cuál permite determinar si el algoritmo al detectar alguna de las acciones realizar por las personas su respuesta es acertada o no, esto con el fin de realizar correcciones en dado caso que el algoritmo implementado detecte las acciones de manera opuesta a lo esperado.

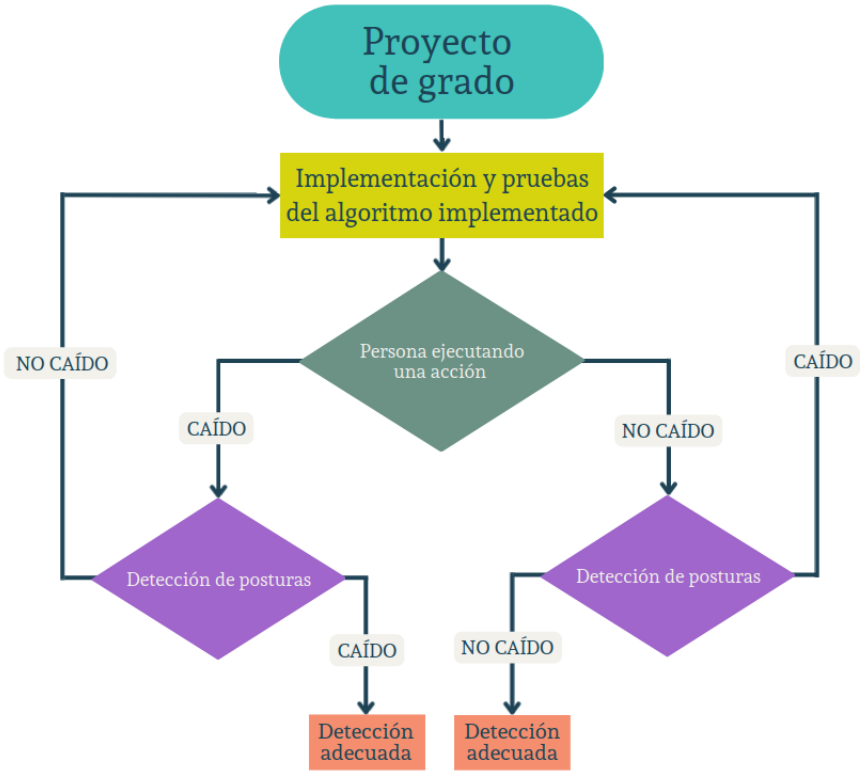


Figura 9.18: Flujo para revisión del algoritmo implementado.

9.5. Actividades para objetivo específico 5

9.5.1. Actividad 5.1 - Revisión final del algoritmo implementado

9.5.1.1. Matriz de confusión

Para la revisión final del algoritmo es menester emplear una matriz de confusión con el objetivo de evidenciar el desempeño del algoritmo implementado, en este caso la idea es comprar los resultados obtenidos a partir de los valores reales contra los valores que se van a predecir [7]. En este caso la matriz de confusión consta de los siguientes 4 opciones que se pueden evidenciar de manera gráfica en la figura 10.1.

- **Verdadero positivo (TP):** Significa que el valor real es positivo y la predicción también es positiva.
- **Verdadero negativo (TN):** Significa que el valor real es negativo y la predicción también es negativa.
- **Falso positivo (FP):** Significa que el valor real es positivo, la predicción es negativa y es error tipo 1.
- **Falso negativo (FN):** Significa que el valor real es negativo, la predicción es positiva y es error tipo 2.

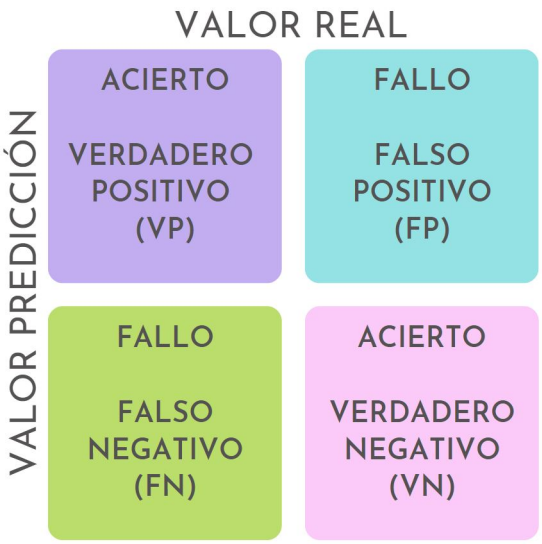


Figura 9.19: Matriz de confusión

Es importante tener presente que los valores que se van a predecir se toman como valores positivos o negativos, mientras que los valores reales se toman como valores verdaderos y falsos.

A partir de las opciones evidenciadas anteriormente, se pueden determinar algunas métricas.

1. **Precisión (PR):** Esta métrica permite determinar la calidad del modelo implementado en la clasificación.

$$PR = \frac{VP}{VP + FP} \tag{9.1}$$

2. **Exhaustividad (RE):** esta métrica indica que cantidad que el modelo puede determinar.

$$RE = \frac{VP}{VP + FN} \tag{9.2}$$

3. **Valor-F (F_1):** Permite determinar el rendimiento tanto de la presición como de la exhaustividad en una sola ecuación.

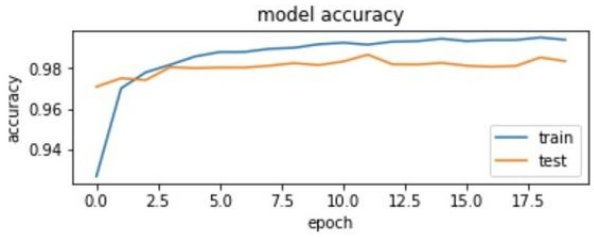
$$F_1 = \frac{2VP}{2VP + FP + FN} \tag{9.3}$$

4. **Exactitud (CA):** Permite determinar el porcentaje de acierto en la detección del modelo.

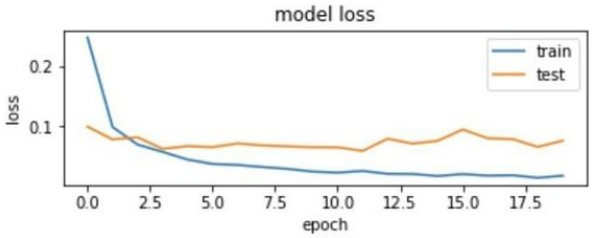
$$CA = \frac{VP + VN}{VP + VN + FP + FN} \tag{9.4}$$

9.5.1.2. Gráficas de exactitud y pérdida

Por otro lado, con ayuda de las gráficas de exactitud y pérdida, es posible evidenciar gráficamente el comportamiento del modelo.



(a) Gráfica de exactitud [59]



(b) Gráfica de pérdida [59]

Figura 9.20: Gráficas de exactitud y pérdida

Como se evidencia en la figura 9.20a, la exactitud cómo se mencionó anteriormente se puede representar como un valor porcentual, pero de igual forma puede ser representado a través de una gráfica que permite monitorear las fases del entrenamiento, por otro lado, en la figura 9.20b, se evidencia la gráfica de pérdidas, que es básicamente la suma de todos los errores que se cometen a lo largo del entrenamiento para los conjuntos de entrenamiento y validación, el objetivo final de este tipo de medida es que se logre minimizar lo máximo posible [59].

Capítulo 10

Resultados

Una vez se realiza la implementación de la red neuronal, es necesario determinar el comportamiento que esta a partir de 3 parámetros distintos como se evidencia a continuación:

1. **Matriz de confusión y reporte de clasificación:**

   Es importante recalcar que los modelos de clasificación manejan métricas que permiten evaluar la calidad del modelo, para este caso existe dos formas que de obtener estos parámetros, el primero es a partir de la generación de una matriz de clasificación, que se evidencia en la figura 10.1, que se obtuvo a partir de los resultados obtenidos del algorimo implementado (Ver anexo 11).

Figura 10.1: Matriz de confusión del algoritmo implementado

Como se evidencia en la sección 9.5.1.1 a partir de los aciertos (verdaderos positivos (VP) y verdaderos negativos (VN)) y de los fallos (falso positivo (FP) y falso negativo (FN)), se obtienes las siguientes métricas:

a) **Precisión (PR)**: Se obtiene de la sección 9.5.1.1 y hace referencia a la ecuación 9.1, se obtiene lo siguiente:

$$PR = \frac{41}{41+0} \tag{10.1}$$

$$PR = 1 \tag{10.2}$$

b) **Exhaustividad (RE)**: Se obtiene de la sección 9.5.1.1 y hace referencia a la ecuación 9.2, se obtiene lo siguiente:

$$PR = \frac{41}{41+0} \tag{10.3}$$

$$PR = 1 \tag{10.4}$$

c) **Valor-F (F_1)**: Se obtiene de la sección 9.5.1.1 y hace referencia a la ecuación 9.3, se obtiene lo siguiente:

$$F_1 = \frac{2*41}{2*41+0+0} \tag{10.5}$$

$$F_1 = 1 \tag{10.6}$$

d) **Exactitud (CA)**: Se obtiene de la sección 9.5.1.1 y hace referencia a la ecuación 9.4, se obtiene lo siguiente:

$$CA = \frac{41+39}{41+39+0+0} \tag{10.7}$$

$$CA = 1 \tag{10.8}$$

De igual forma es posible obtener un reporte de clasificación que nos permite evidenciar 3 de las 4 métricas expuestas anteriormente que son la presición, la exhaustividad y el valor-F, pero en este caso a diferencia de la matriz de confusión se hace el cálculo de los parámetros para las dos acciones a detectar, este reporte se evidencia en la figura 10.4 (Ver anexo 11).

	precision	recall	f1-score	support
NO CAIDA	1.00	1.00	1.00	41
CAIDA	1.00	1.00	1.00	39
accuracy			1.00	80
macro avg	1.00	1.00	1.00	80
weighted avg	1.00	1.00	1.00	80

Figura 10.2: Reporte de clasificación del algoritmo implementado

2. **Gráficas de exactitud y pérdida:**

   Para el tema de las gráficas se obtiene loas siguientes gráficas de nos permiten evidenciar la relación entre los aciertos y las pérdidas del modelo obtenidos a lo largo de cada época (Ver anexo 11):

   - **Gráfica de aciertos:**

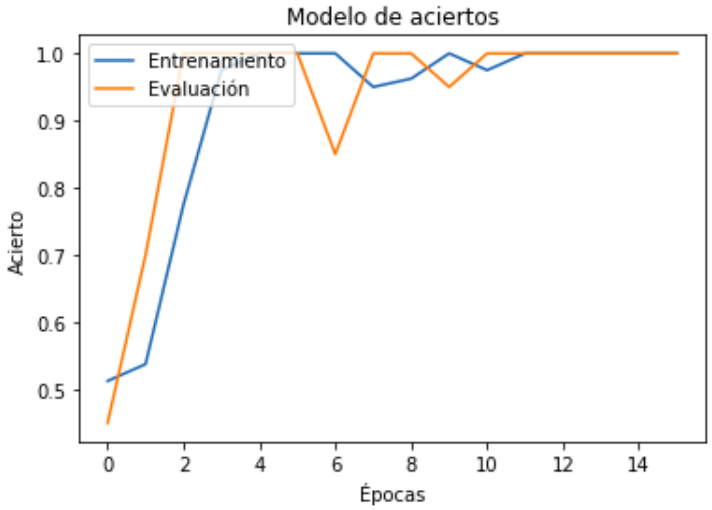


Figura 10.3: Gráfica de aciertos del modelo

   - **Gráfica de pérdidas**

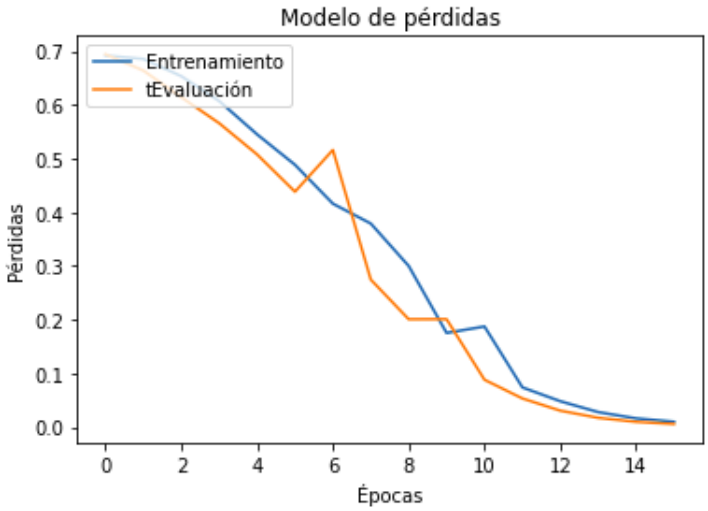


Figura 10.4: Gráfica de pérdidas del modelo

3. **Pruebas en tiempo real:**

   Se realizan pruebas en tiempo real con diferentes personas ejecutando las dos acciones a detectar, a continuación se evidenciar algunas fotografías, cabe aclarar que las personas que aparecen en las fotos gozan de excelentes condiciones de salud (Ver anexo 11).

(a) No caída

(b) Caída

(c) Caída

(d) No caída

(e) Caída

(f) No caída

(g) Caída

(h) No caída

(i) No caída

Figura 10.5: Pruebas en tiempo real

Capítulo 11

Conclusiones y Trabajos Futuros

- A partir de la revisión bibliográfica, se determinó que MediaPipe es una herramienta de aprendizaje de máquina robusta y potente para la identificación y la interpretación de puntos específicos en el cuerpo, lo que facilita la identificación de poses rápida y efectivamente. Particularmente para este trabajo se prefirió la Utilidad Holistic de MediaPose, debido a que está optimizada para la detección en cara, manos y poses, adicionalmente es más liviana permitiendo que no sea necesario contar con un hardware muy robusto para su funcionamiento.
- Al comparar el comportamiento de las redes recurrentes y las redes LSTM en el entrenamiento del sistema, se encontró que las últimas son más adecuadas para este proyecto ya que al procesar video en tiempo real era necesario procesar secuencias más grandes aspecto que se dificulta en las redes recurrentes.
- Se evidencia que el acierto que tuvo el modelo implementado fue considerablemente alto, considerando la cantidad de parámetros que se tomaron para realizar el entrenamiento.
- Dentro de las pruebas desarrolladas en el desarrollo de este trabajo se observó que la detección de caídas cumplió cabalmente para una persona, sin embargo como trabajo futuro sería interesante estudiar qué pasaría con la detección para múltiples personas por parte del sistema desarrollado.
- Este proyecto a futuro puede ser implementado en un robot de asistencia, dada la funcionalidad que tiene el algoritmo desarrollado, para que pueda detectar las caídas cuando las personas se movilizan en un entorno cerrado, generando alertas en caso de ocurrir.

Bibliografía

[1] Ahmed Abobakr, Mohammed Hossny, Hala Abdelkader, and Saeid Nahavandi. Rgb-d fall detection via deep residual convolutional lstm networks. In *2018 Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–7, 2018.

[2] Charu C Aggarwal et al. Neural networks and deep learning. *Springer*, 10:978–3, 2018.

[3] Xavier Basogain. Redes neuronales artificiales y sus aplicaciones. *Dpto. Ingeniería de Sistemas y Automática, Escuela Superior de Ingeniería Bilbao, howpublished=http://ocw.ehu.es/ensenanzas-tecnicas/redes-neuronales-artificiales-y-sus-aplicaciones/Course_ listing., year=2008.*

[4] Valentin Bazarevsky and Google Research Ivan Grishchenko, Research Engineers. On-device, real-time body pose tracking with mediapipe blazepose. `https://ai.googleblog.com/2020/08/on-device-real-time-body-pose-tracking.html`, aug 2020.

[5] Avijeet Biswal. Convolutional neural network tutorial. `https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network`, jul 2022.

[6] Avijeet Biswal. Recurrent neural network (rnn) tutorial: Types, examples, lstm and more. `https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn`, aug 2022.

[7] Sebastian Bittrich, Marika Kaden, Christoph Leberecht, Florian Kaiser, Thomas Villmann, and Dirk Labudde. Application of an interpretable classification model on early folding residues during protein folding. *BioData mining*, 12(1):1–16, 2019.

[8] Harsha Bommana. Introduction to neural networks — part 1. `https://medium.com/deep-learning-demystified/introduction-to-neural-networks-part-1-e13f132c6d7e`, may 2019.

[9] Kabalan Chaccour, Rony Darazi, Amir Hajjam El Hassani, and Emmanuel Andrès. From fall detection to fall prevention: A generic classification of fall-related systems. *IEEE Sensors Journal*, 17(3):812–822, 2017.

[10] Madhav Chamle, K. G. Gunale, and K. K. Warhade. Automated unusual event detection in video surveillance. In *2016 International Conference on Inventive Computation Technologies (ICICT)*, volume 2, pages 1–4, 2016.

[11] Pongsatorn Chutimawattanakul and Pranchalee Samanpiboon. Fall detection for the elderly using yolov4 and lstm. In *2022 19th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pages 1–5, 2022.

[12] Intel Corporation. Open source computer vision. https://docs.opencv.org/4.6.0/, jun 2022.

[13] DANE. Adulto mayor en colombia - características generales. https://www.dane.gov.co/files/investigaciones/genero/presentacion-caracteristicas-generales-adulto-mayor-en-colombia.pdf, jan 2021.

[14] Gobierno de Colombia. Misión de sabios - colombia - 2019. https://minciencias.gov.co/sites/default/files/libro_mision_de_sabios_digital_1_2_0.pdf, dec 2021.

[15] Congreso de la República. Medidas de protección al adulto mayor en colombia. http://www.secretariasenado.gov.co/senado/basedoc/ley_1850_2017.html, jul 2017.

[16] Congreso de la República. ConvenciÓn interamericana sobre la protecciÓn de los derechos humanos de las personas mayores. https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=141981, sep 2020.

[17] Sonia Patricia de Santillana Hernández, Luis Eduardo Alvarado Moctezuma, Gustavo Rodrigo Medina Beltrán, Gricelda Gómez Ortega, Rosa María Cortés Gonzálezzuela Espinoza, and Rosa María Cortés González. Caídas en el adulto mayor. factores intrínsecos y extrínsecos. *Revista Médica del Instituto Mexicano del Seguro Social*, 40(6):489–493, 2002.

[18] Riya Dichwalkar, Shreya Oak, Tania Rajabally, and Dhananjay Kalbande. Activity recognition and fall detection in elderly people. In *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–6, 2020.

[19] Tran Duc Tan and Nguyen Van Tinh. Reliable fall detection system using an 3-dof accelerometer and cascade posture recognitions. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*, pages 1–6, 2014.

[20] Nashwa El-Bendary, Qing Tan, Frédérique C Pivot, and Anthony Lam. Fall detection and prevention for the elderly: A review of trends and challenges. *International Journal on Smart Sensing & Intelligent Systems*, 6(3), 2013.

[21] National Geografic España. Sinapsis, el lenguaje de las neuronas en el cerebro. https://n9.cl/6vjro9, apr 2019.

[22] H Espínola. Caídas en el adulto mayor. *Boletín de la pontificia Universidad Católica de Chile*, 29:1–5, 2000.

[23] M.M. Fadoul. *Python Machine Learning: Machine Learning and Deep Learning from Scratch Illustrated with Python, Scikit-Learn, Keras, Theano and Tensorflow.* Independently Published, 2020.

[24] Yaxiang Fan, Martin D. Levine, Gongjian Wen, and Shaohua Qiu. A deep neural network for real-time detection of falling humans in naturally occurring scenes. *Neurocomputing*, 260:43–58, 2017.

[25] Aditya Rio Fansdana, Aulia Khamas Heikhmakhtiar, and Satria Mandala. Real-time falling detection system for elderly using cnn. In *2021 International Conference on Data Science and Its Applications (ICoDSA)*, pages 194–197, 2021.

[26] Shubham Garg, Aman Saxena, and Richa Gupta. Yoga pose classification: a cnn and mediapipe inspired deep learning approach for real-world application. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–12, 2022.

[27] Sparsh Gupta. The 7 most common machine learning loss functions. `https://builtin.com/machine-learning/common-loss-functions`, apr 2022.

[28] Khadija Hanifi and M. Elif Karsligil. Elderly fall detection with vital signs monitoring using cw doppler radar. *IEEE Sensors Journal*, 21(15):16969–16978, 2021.

[29] Md Mahedi Hasan, Md Shamimul Islam, and Sohaib Abdullah. Robust pose-based human fall detection using recurrent neural network. In *2019 IEEE International Conference on Robotics, Automation, Artificial-intelligence and Internet-of-Things (RAAICON)*, pages 48–51, 2019.

[30] Kyaw Kyaw Htike, Othman O. Khalifa, Huda Adibah Mohd Ramli, and Mohammad A. M. Abushariah. Human activity recognition for video surveillance using sequences of postures. In *The Third International Conference on e-Technologies and Networks for Development (ICeND2014)*, pages 79–82, 2014.

[31] Madoka Inoue, Ryo Taguchi, and Taizo Umezaki. Bed-exit prediction applying neural network combining bed position detection and patient posture estimation. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 3208–3211, 2019.

[32] Shruti Jadon. What is an activation function? what are commonly used activation functions? `https://medium.com/@shrutijadon/survey-on-activation-functions-for-deep-learning-9689331ba092`, mar 2018.

[33] Rose A Kenny, Cliodhna Ni Scanaill, and Michael McGrath. Falls prevention in the home: Challenges for new technologies. In *Intelligent Technologies for Bridging the Grey Digital Divide*, pages 46–64. IGI Global, 2011.

[34] Sangyeon Kim, Gawon Lee, and Jihie Kim. Lightweight real-time fall detection using bidirectional recurrent neural network. In *2020 Joint 11th International Conference on Soft Computing and Intelligent Systems and 21st International Symposium on Advanced Intelligent Systems (SCIS-ISIS)*, pages 1–5, 2020.

[35] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

[36] Xuan-Hien Le, Hung Viet Ho, Giha Lee, and Sungho Jung. Application of long short-term memory (lstm) neural network for flood forecasting. *Water*, 11(7):1387, 2019.

[37] Xiaogang Li, Tiantian Pang, Weixiang Liu, and Tianfu Wang. Fall detection for elderly person care using convolutional neural networks. In *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 1–6, 2017.

[38] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014.

[39] Google LLC. Mediapipe pose. https://google.github.io/mediapipe/solutions/pose.html, jun 2020.

[40] Patricio Loncomilla. Deep learning: Redes convolucionales. *Recuperado de https://ccc. inaoep. mx/˜ pgomez/deep/presentations*, 2016.

[41] Steven T Moore, Valentina Dilda, Bandar Hakim, and Hamish G MacDougall. Validation of 24-hour ambulatory gait assessment in parkinson's disease with simultaneous video observation. *Biomedical engineering online*, 10(1):1–9, 2011.

[42] Sarah Mroz, Natalie Baddour, Connor McGuirk, Pascale Juneau, Albert Tu, Kevin Cheung, and Edward Lemaire. Comparing the quality of human pose estimation with blazepose or openpose. pages 1–4, 12 2021.

[43] Abdulmajid Murad and Jae-Young Pyun. Deep recurrent neural networks for human activity recognition. *Sensors*, 17(11):2556, 2017.

[44] Brian Mwandau and Matunda Nyanchama. *Investigating Keystroke Dynamics as a Two-Factor Biometric Security*. PhD thesis, 06 2018.

[45] Cristobal Olah. Introducción a la memoria a largo corto plazo (lstm). https://colah.github.io/posts/2015-08-Understanding-LSTMs/, aug 2015.

[46] Artem Oppermann. Loss functionsin deep learning. jun 2021.

[47] Jesús Pérez Guerrero. Redes recurrentes. 2020.

[48] M Gestal Pose. Introducción a las redes de neuronas artificiales. *Departamento de Tecnologías de la Información y las Comunicaciones. Universidad da Coruña*, 2009.

[49] Patryk Radzki. Detection of human body landmarks - mediapipe and openpose comparison. https://www.hearai.pl/post/14-openpose/, apr 2022.

[50] Nicholas Renotte. Action detection for sign language. https://github.com/nicknochnack/ActionDetectionforSignLanguage, jun 2021.

[51] Laura María Álvarez Rodríguez. Síndrome de caídas en el adulto mayor. *Revista Médica de Costa Rica y Centroamérica*, 72(617):807–810, 2016.

[52] Ravi Kiran Selvam. Adagrad and adadelta optimizers - implementation and testing:-. https://www.sravikiran.com/GSOC18/page2/, jul 2018.

[53] N Senthilkumar, M Manimegalai, S Karpakam, S.R Ashokkumar, and M Premkumar. Human action recognition based on spatial–temporal relational model and lstm-cnn framework. *Materials Today: Proceedings*, 57:2087–2091, 2022. International Conference on Innovation and Application in Science and Technology.

[54] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *towards data science*, 6(12):310–316, 2017.

[55] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[56] simpl!learn. What is neural network: Overview, applications, and advantages. https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-neural-network, jan 2022.

[57] Ruoyu Sun. Optimization for deep learning: theory and algorithms. *arXiv preprint arXiv:1912.08957*, 2019.

[58] M. Taylor. *Make Your Own Neural Network: An In-Depth Visual Introduction for Beginners*. Amazon Digital Services LLC - KDP Print US, 2017.

[59] tutorialspoint. Evaluating model performance. https://www.tutorialspoint.com/deep_learning_with_keras/deep_learning_with_keras_evaluating_model_performance.htm#, feb 2022.

[60] Pavan Vadapalli. Biological neural network: Importance, components & comparison. https://www.upgrad.com/blog/biological-neural-network/, feb 2021.

[61] S Yan. Understanding lstm networks. *Online). Accessed on August*, 11, 2015.

[62] Xiaokang Zhou, Wei Liang, Kevin I-Kai Wang, Hao Wang, Laurence T. Yang, and Qun Jin. Deep-learning-enhanced human activity recognition for internet of healthcare things. *IEEE Internet of Things Journal*, 7(7):6429–6438, 2020.

ANEXOS

1. Importar librerías

```
import cv2                                  # Libreria de visi n artificial
import numpy as np                          # Libreria para calculo numerico
import os                                   # Modulo para trabajar con so
from matplotlib import pyplot as plt        # Libreria de visualizacion de datos
import time                                 # Funcion de temporizador
import mediapipe as mp                      # Modelos de ML para deteccion
```

2. Determinar puntos de referencia con MediaPipe Holistic.

- Variables

```
mp_holistic = mp.solutions.holistic         # Modelo Holistic
mp_drawing = mp.solutions.drawing_utils     # Utilidades de dibujo para los
    puntos de referencia
```

Función 1: Detección de imagen.

```
def mediapipe_detection(image, model):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # Convertir imagen de BGR a
    RGB
    image.flags.writeable = False                  # Imagen no apta para uso
    results = model.process(image)                 # Prediccion
    image.flags.writeable = True                   # Imagen apta para uso
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR) # Convertir imagen de RGB a
    BGR
    return image, results
```

Función 2: Dibujar punto de referencia.

```
def draw_landmarks(image, results):
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.
    POSE_CONNECTIONS)       # Conexiones de todo el cuerpo
```

Función 3: Dibujar punto de referencia con colores.

```
def draw_styled_landmarks(image, results): # Hacer las lineas con colores
    determinados
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.
    POSE_CONNECTIONS,                   # Dibujar conexiones de todo el cuerpo
                              mp_drawing.DrawingSpec(color=(51, 40, 28),
    thickness = 4, circle_radius = 4),
                              mp_drawing.DrawingSpec(color=(176, 201, 72),
    thickness = 4, circle_radius = 2))
```

Toma de vídeo y detección de puntos de referencia.

```
cap = cv2.VideoCapture(2)
# Acceso al modelo Holistic
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence
    =0.5) as holistic:
     while cap.isOpened():
         # Lectura de video
         ret, frame = cap.read()
         # Predicciones
         image, results = mediapipe_detection(frame, holistic)
         print(results)
         # Dibujo de los puntos de referencia
         draw_styled_landmarks(image, results)
         # Visualizacion en la pantalla
         cv2.imshow('OpenCV Feed', image)
         # Cierre de la ventana
         if cv2.waitKey(10) & 0xFF == ord('q'):
             break
     cap.release()
     cv2.destroyAllWindows()
```

Prueba de toma de resultados.

```
draw_landmarks(frame, results)
plt.imshow(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
```

3. Extracción de los valores de los puntos de referencia.
Convertir datos de los resultados en arreglos de tipo Numpy

```
results.pose_landmarks
```

Concatenar y extraer los datos.
Función 4: Extraer puntos de referencia.

```
def extract_keypoints(results):
    # Son 3 valores (x, y, z) y 4 para pose (visivility) - res.x, res.y, res.z,
     res.visibility -> Cada uno es un valor individual de cada punto de
    referencia
     pose = np.array([[res.x, res.y, res.z, res.visibility] for res in results.
    pose_landmarks.landmark]).flatten() if results.pose_landmarks else np.zeros
    (33*4)
     # Concatenar todos los puntos
     return np.concatenate([pose])
```

Prueba puntos guardados en .npy

```
result_test = extract_keypoints(results)
result_test

np.save('prueba', result_test)
np.load('prueba.npy')
```

4. Creación de las carpetas para la recolección de datos.
Determinar parámetros y valores.

```
# Ruta para los datos
DATA_PATH = os.path.join('DATOS_MEDIAPIPE_PRUEBA_201122_final_7') # Se crea una
     carpeta con las 2 acciones a detectar y se generan los 50 videos
# Acciones que se van a detectar
actions = np.array(['NO CAIDA', 'CAIDA',])
# 50 videos para cada valor de datos
```

```
no_sequences = 50
# Cada video va a contener una duraci n de 50 frames
sequence_length = 50
```

Creación de cada folder para cada acción con sus respectivos frames y cantidad de vídeos

```
# Un folder por accion
for action in actions:
    for sequence in range(no_sequences):
        try:
            os.makedirs(os.path.join(DATA_PATH, action, str(sequence)))
        except:
            pass
```

5. Recolección de valores de referencia para entrenamiento y evaluación

```
cap = cv2.VideoCapture(2)
# Acceso al modelo Holistic
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence
    =0.5) as holistic:
    # Primer bucle
    # Bucle para las acciones
    for action in actions: # Bucle para todas las acciones
        for sequence in range(no_sequences): # Bucle para las secuencias de los
    videos
            for frame_num in range(sequence_length): # Bucle para la secuencia
    de duracion de los videos

                # Leer video
                ret, frame = cap.read()
                # Predicciones
                image, results = mediapipe_detection(frame, holistic)
                print(results)
                # Dibujo de puntos de referencia
                draw_styled_landmarks(image, results)
                # Logica de espera para tomar los v deos
                if frame_num == 0:
                    cv2.putText(image, 'Starting Collection', (120, 200),
                    cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 4, cv2.LINE_AA)
                    cv2.putText(image, 'Collecting frames for {} video number
    {}'.format(action, sequence), (15, 12),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)
                    # Show to screen
                    cv2.imshow('OpenCV Feed', image)
                    cv2.waitKey(2000)
                else:
                    cv2.putText(image, 'Collecting frames for {} video number
    {}'.format(action, sequence), (15, 12),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)
                    cv2.imshow('OpenCV Feed', image)
                # Exportar puntos de referencia
                keypoints = extract_keypoints(results)
                npy_path = os.path.join(DATA_PATH, action, str(sequence), str(
    frame_num))
                np.save(npy_path, keypoints)
                # Cierre de la ventana
                if cv2.waitKey(10) & 0xFF ==ord('q'):
                    break
```

```
cap.release()
cv2.destroyAllWindows()
```

6. Procesamiento de los datos y creación de las etiquetas.

Importar librerías para el modelo.

```
from sklearn.model_selection import train_test_split # Particion de
    entrenamiento y evaluacion
from tensorflow.keras.utils import to_categorical # Convertir datos en linea
```

Determiar las etiquetas

```
label_map = {label:num for num, label in enumerate(actions)}
label_map
```

Creación de arreglo con los 132 puntos de referencia

```
sequences, labels = [], []
for action in actions:
    for sequence in range(no_sequences):
        window = []
        for frame_num in range(sequence_length):
            res = np.load(os.path.join(DATA_PATH, action, str(sequence), "{}.
    npy".format(frame_num)))
            window.append(res)
        sequences.append(window)
        labels.append(label_map[action])
```

Número de etiquetas, secuencias y puntos de referencia.

```
np.array(sequences).shape

np.array(labels).shape

X = np.array(sequences)
X.shape

y = to_categorical(labels).astype(int)
y.shape
```

División de datos (entrenamiento y evaluación).

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

X_test.shape

X.shape[2]

X_train.shape

y_test.shape

y_train.shape
```

7. Elaboración y entrenamiento de la Red Neuronal LSTM.

Importar librerías para redes neuronales.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.callbacks import TensorBoard
```

Creación de la Red Neuronal LSTM.

```
model  = Sequential()
model.add(LSTM(units = 64, return_sequences = True, input_shape = (X.shape[1], X.shape[2])))
model.add(Dropout(0.2))
model.add(LSTM(units = 128, return_sequences = True))
model.add(Dropout(0.2))
model.add(LSTM(units = 64, return_sequences = True))
model.add(Dropout(0.2))
model.add(LSTM(units = 32))
model.add(Dropout(0.2))

model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(32, activation='relu'))

model.add(Dense(actions.shape[0], activation="sigmoid"))

model.compile(optimizer='Adam', loss='binary_crossentropy', metrics=['accuracy'])

history = model.fit(X_train, y_train, epochs=16, batch_size=32,validation_data=(X_test, y_test))
```

Resumen del modelo.

```
model.summary()
```

8. Predicciones.

```
res = model.predict(X_test)

actions[np.argmax(res[0])]

actions[np.argmax(y_test[0])]
```

9. Almacenamiento de los pesos.

```
model.save('modelo_tesis_mpsaab_final_201122_mami.h5')
```

10. Matriz de confusión, gráficas y reporte de clasificación.

Matriz de confusión.

```
from sklearn.metrics import multilabel_confusion_matrix, accuracy_score, confusion_matrix

yhat = model.predict(X_train)

ytrue = np.argmax(y_train, axis=1).tolist()
yhat = np.argmax(yhat, axis=1).tolist()

multilabel_confusion_matrix(ytrue, yhat)

cf_matrix = confusion_matrix(ytrue, yhat)

import seaborn as sns

ax = sns.heatmap(cf_matrix, annot=True, cmap='Blues')
```

```
15
16 ax.set_title('Matriz de confusion con etiquetas');
17 ax.set_xlabel('Valores predecidos')
18 ax.set_ylabel('Valores actuales');
19
20 ## Nombre de las etiquetas
21 ax.xaxis.set_ticklabels(['Falso','Verdadero'])
22 ax.yaxis.set_ticklabels(['Verdadero','Falso'])
23
24 ## Visualizar
25 plt.show()
26
27 accuracy_score(ytrue, yhat)
```

Reporte de clasificación.

```
1 from sklearn.metrics import classification_report
2 print(classification_report(ytrue, yhat, target_names=label_map))
```

Gráficas de exactitud y pérdida.

```
1 # Graficas de modelo de acierto y perdidas
2 # Aciertos
3 plt.plot(history.history['accuracy'])
4 plt.plot(history.history['val_accuracy'])
5 plt.title('Modelo de aciertos')
6 plt.ylabel('Acierto')
7 plt.xlabel('Epocas')
8 plt.legend(['Entrenamiento', 'Evaluacion'], loc='upper left')
9 plt.show()
10 # Perdidas
11 plt.plot(history.history['loss'])
12 plt.plot(history.history['val_loss'])
13 plt.title('Modelo de p rdidas')
14 plt.ylabel('P rdidas')
15 plt.xlabel(' pocas ')
16 plt.legend(['Entrenamiento', 'EvaluaciOn'], loc='upper left')
17 plt.show()
```

11. Prueba en tiempo real.

```
1 colors = [(0, 255, 97), (255, 76, 255)]
2 def prob_viz(res, actions, input_frame, colors):
3     output_frame = input_frame.copy()
4     for num, prob in enumerate(res):
5         cv2.rectangle(output_frame, (0, 60 + num * 40), (int(prob * 100), 90 +
    num * 40), colors[num], -1)
6         cv2.putText(output_frame, actions[num], (0, 85 + num * 40), cv2.
    FONT_HERSHEY_SIMPLEX, 1, (255, 0, 255), 2, cv2.LINE_AA)
7
8     return output_frame
9
10     from keras.models import load_model
11 model = load_model('modelo_tesis_mpsaab_final_201122_mami.h5')
12
13 # Variables a detectar
14 sequence = [] # Concatenar datos
15 sentence = []
16 predictions = []
```

```
threshold = 0.5

cap = cv2.VideoCapture(2)
# Acceder a modelo holisitico - mediapipe model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():
        # Lectura
        ret, frame = cap.read()
        # Detecciones
        image, results = mediapipe_detection(frame, holistic)
        print(results)
        # Dibujar lineas
        draw_styled_landmarks(image, results)
        # 2. Predicciones
        keypoints = extract_keypoints(results)
        #
        sequence.append(keypoints)
        sequence = sequence[-50:]

        if len(sequence) == 50:
            res = model.predict(np.expand_dims(sequence, axis=0))[0]
            print(actions[np.argmax(res)])
            predictions.append(np.argmax(res))

        # 3. Visualizar
            if np.unique(predictions[-10:])[0] == np.argmax(res):
                if res[np.argmax(res)] > threshold:
                    if len(sentence) > 0:
                        if actions[np.argmax(res)] != sentence[-1]: # Acciones cambian
                            sentence.append(actions[np.argmax(res)])
                    else:
                        sentence.append(actions[np.argmax(res)])
                if len(sentence) > 5:
                    sentence = sentence[-5:]

            # Probabilidades
            image = prob_viz(res, actions, image, colors)

        cv2.rectangle(image, (0,0), (640, 40), (169, 204, 227), -1)
        cv2.putText(image, ' '.join(sentence), (3, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (231, 76, 60), 2, cv2.LINE_AA)

        # Ver pantalla
        cv2.imshow("OpenCV Feed", image)
        # Break
        if cv2.waitKey(10) & 0xFF ==ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()
```