Diseño e implementación de un sistema de detección de caídas en el adulto mayor mediante visión artificial utilizando redes neuronales convolucionales (deep learning)

Orbe Cisneros, Edwin Alexander

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Trabajo de titulación, previo a la obtención del título de Ingeniero en Electrónica, Automatización y Control

Ing. Vargas Vallejo, Vanessa Carolina, PhD

01 de agosto del 2022

COPYLEAKS

Trabajo Titulacion - Alexander Orbe.pdf

Scanned on: 18:46 August 2, 2022 UTC

firmado electrónicamente por:
VANESSA CAROLINA
VARGAS VALLEJO

6.2%	41	17229
Overall Similarity Score	Results Found	Total Words in Text

Identical Words	499
Words with Minor Changes	129
Paraphrased Words	448
Omitted Words	0

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Certificación

Certifico que el trabajo de titulación: "**Diseño e implementación de un sistema de detección de caídas en el adulto mayor mediante visión artificial utilizando redes neuronales convolucionales (deep learning)**" fue realizado por el señor **Orbe Cisneros, Edwin Alexander;** el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizado en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Sangolquí, 01 de agosto del 2022

Firma:

Firmado electrónicamente por:
VANESSA CAROLINA
VARGAS VALLEJO

.....

Ing. Vargas Vallejo, Vanessa Carolina, PhD.

C. C. 1711309045

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Responsabilidad de Autoría

Yo, **Orbe Cisneros, Edwin Alexander**, con cédula de ciudadanía n°0401318712, declaro que el contenido, ideas y criterios del trabajo de titulación: **Diseño e implementación de un sistema de detección de caídas en el adulto mayor mediante visión artificial utilizando redes neuronales convolucionales (deep learning)** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 01 de agosto del 2022

Firma

.....

Orbe Cisneros, Edwin Alexander

C.C.: 0401318712

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Autorización de Publicación

Yo **Orbe Cisneros, Edwin Alexander**, con cédula de ciudadanía n°0401318712, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **Diseño e implementación de un sistema de detección de caídas en el adulto mayor mediante visión artificial utilizando redes neuronales convolucionales (deep learning)** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 01 de agosto del 2022

Firma

.....

Orbe Cisneros, Edwin Alexander

C.C.: 0401318712

DEDICATORIA

Primeramente, quiero dedicar este trabajo al padre de toda la creación, Dios. Quien ha llenado mi vida de sabiduría. A la virgen María y todos los santos que me han acompañado en este camino recorrido.

A mis padres Edwin y Magola que con amor me educaron y brindaron consejos para enfrentarme a las adversidades de la vida y me han apoyado en todo momento.

A mi hija Anahí quien me enseñó a ser padre y el motivo más lindo para superarme todos los días.

De igual forma, dedico esta tesis a mi abuelita quien ha sido un pilar importante en mi existencia. Abuelita sé que desde el cielo iluminas mi sendero y eres parte importante de estos logros alcanzados.

Edwin Alexander Orbe Cisneros

AGRADECIMIENTO

De manera muy especial quiero expresar mi gratitud a Dios, por enseñarme a ser fuerte en los tiempos difíciles y agradecido en las metas conseguidas.

Mi aprecio y agradecimiento a la Universidad de las Fuerzas Armadas ESPE, por abrirme sus puertas del conocimiento para formarme en el transcurso de estos años.

A mi Directora de Tesis por brindarme la oportunidad de realizar este trabajo de titulación, por sus enseñanzas, por la paciencia y por el tiempo dedicado.

A mis padres que con su apoyo moral y económico han sabido guiarme por el camino correcto y darme esta linda herencia que es el estudio.

Finalmente quiero expresar mi agradecimiento a mis profesores, amigos y familiares que de alguna manera aportaron su granito de arena para el desarrollo de este trabajo.

Edwin Alexander Orbe Cisneros

Índice de Contenidos

Índice de Tablas

Índice de figuras

Resumen

El presente proyecto de titulación consiste en el desarrollo de un sistema de detección de caídas en el adulto mayor, utilizando técnicas de aprendizaje profundo. Para el entrenamiento del sistema de detección se usan dos sets de datos con imágenes de las actividades diarias de varias personas. El primer set de datos es el *UR Fall Detection* Dataset que puede ser descargado de la web, en tanto que el segundo Dataset fue creado por el autor de este trabajo. Este último contiene fotografías de personas en diferentes posiciones como parada, sentada, agachada, acostada y caída. Las imágenes correspondientes al conjunto de datos final se etiquetan utilizando la herramienta *LabelImg*. LabelImg es un software de anotación de imágenes que permite marcar regiones de interés con las clases de objeto correspondientes. Una vez etiquetado todo el set de datos se procede al entrenamiento del sistema de detección de caídas mediante la red neuronal convolucional (CNN). Para este propósito, se aplica el concepto de Aprendizaje por transferencia, que permite trabajar con CNN ya entrenadas y capacitarlas para solucionar un nuevo problema en específico. El modelo de CNN con el que se trabaja es SSD *Mobilenet v2*. Este es un modelo de red neuronal profunda implementado en equipos de bajas características computaciones, como son los dispositivos móviles (de allí el nombre *Mobilenet*), y con un rendimiento de alta precisión en detección de objetos. Finalmente se procede a realizar las pruebas del modelo de CNN mediante la detección de objetos de clase en escenas capturadas en tiempo real, mediante una cámara integrada al computador.

Palabras Clave: Aprendizaje profundo, LabelImg, red neuronal convolucional, Aprendizaje por transferencia, SSD Mobilenet v2.

Abstract

The present research project consists of the development of a system for the detection of falls in the elderly, using deep learning techniques. Two datasets are used to train the detection system, which contains images of daily activities of several people. The first dataset is the UR Fall Detection Dataset that can be downloaded from the web, while the second one is created by the author of this work. The latter contains pictures of people in different poses such as standing, sitting, crouching, lying down and fallen. The images corresponding to the final dataset are labeled in the *LabelImg* tool. LabelImg is a software, which is an image annotation software that allows marking regions of interest with the corresponding object classes, was used. Once the whole dataset is labeled, the next step is the training of the fall detection system using the convolutional neural network (CNN). For this purpose, the concept of Transfer Learning is applied. It allows working with already trained CNNs and training them to solve a specific new problem. The CNN model used is SSD *Mobilenet* v2, a deep neural network model implemented in equipment with low computational characteristics, such as mobile devices (hence the name *Mobilenet*), and with high precision performance in object detection. Finally, the CNN model is tested by detecting class objects in scenes captured in real time, using a camera integrated to the computer.

Key words: deep learning, LabelImg, convolutional neural network, Transfer Learning, SSD Mobilenet v2.

Capítulo I. Introducción

Antecedentes

Como es ampliamente conocido, las personas adultas mayores con el pasar del tiempo presentan problemas con su salud, deteriorándose gradualmente su capacidad física y mental. La pérdida del equilibrio es bastante frecuente lo que ocasiona accidentes de caídas que pueden ser catastróficos. Según la Organización Mundial de la Salud (OMS), las caídas representan un problema mundial de salud pública. La OMS estima que anualmente fallecen en todo el mundo unas 684 000 personas debido a caídas; convirtiéndolas en la segunda causa de muerte a nivel mundial provocada por traumatismos involuntarios (Organización Mundial de la Salud, 2021).

En el Ecuador, el Instituto Nacional de Estadísticas y Censos (INEC) realizó una encuesta para conocer el estado de salud y aspectos relacionados al envejecimiento en personas adultos mayores. Una de las áreas encuestadas fue precisamente el número de caídas registradas en el periodo de estudio (2009 - 2010) por personas mayores a 60 años. El 37.4% de los encuestados sufrieron caídas, como se muestra en la Tabla 1 (Freire, 2009 -2010).

Tabla 1

Caídas en el año 2009 - 2010 por sexo y grupos de edad. (Freire, 2009 -2010)

Edad (años)	Mujeres que sufrieron caídas %	Hombres que sufrieron caídas %	Total, Hombres y Mujeres %
60 a 64	37.1	24.6	31.2
65 a 74	46.3	29.8	38.7
75 o más	44.4	36.4	40.6
Todos	43.3	30.6	37.4

Motivación del proyecto

Considerando la importancia de esta problemática, se puede evidenciar en la literatura varios estudios enfocados en determinar la detección de caídas en este grupo de personas vulnerables. Con el objetivo de generar algún tipo de alerta cuando se presenten posibles eventos que pongan en riesgo la salud del adulto mayor (García Crespo, Rodríguez Goncalvez, Garcés, & García Tejedor, s.f.). En el estudio del estado del arte se pueden verificar principalmente dos tipos de sistemas: los basados en tecnologías con sensores y los que utilizan inteligencia artificial (IA).

Por una parte, las tecnologías con sensores proporcionan información relacionada con el movimiento y entorno del ser humano. En cuanto al movimiento se analizan parámetros como posición y velocidad que son tomados directamente de la persona. En tanto que, al entorno del ser humano se toma información referente a la ausencia o presencia de la persona y la abertura o cierre de puertas en un ambiente determinado (Izama Flores, 2017) (Pérolle & Etxeberria Arritxabal, 2017). Otros estudios referidos a la detección de caídas utilizan técnicas de Inteligencia Artificial (González Díaz, 2017). Las cuales se clasifican en: Visión artificial y aprendizaje profundo.

La visión artificial se basa en el uso de algoritmos de análisis estadístico en imágenes (Diaz Sola, 2015). Donde las imágenes pueden ser representadas mediante una división jerárquica de varios niveles o capas, representadas con píxeles (Galeote Gutiérrez, 2018). La estimación por niveles o capas permite separar el área de fondo, de las zonas de mayor interés de la imagen. De esta forma se puede identificar posibles eventos asociados a la caída de una persona. Cabe mencionar que los algoritmos de visión artificial tienen algunas desventajas como: su coste computacional y la capacidad de detección de personas u objetos, la cual se ve seriamente afectada en ambientes sujetos a cambios de iluminación.

En efecto, para afrontar estos inconvenientes surgen otro tipo de técnicas dentro de lo que se conoce como aprendizaje profundo o deep learning. El mismo que se constituye de una arquitectura constituida por capas formadas por neuronas, conocidas como redes neuronales artificiales. De ahí su nombre “aprendizaje profundo” referido al uso de múltiples capas de red. Además, las redes neuronales artificiales imitan las características arquitecturales del sistema nervioso del ser humano. Haciendo que estas redes se especialicen en el desarrollo de tareas relacionadas con la visión como la detección y el reconocimiento de formas, objetos y personas (Matich, 2001).

Dentro de este grupo de técnicas, existe un tipo específico de redes neuronales artificiales conocidas como redes neuronales convolucionales (CNN), diseñadas especialmente para trabajar con imágenes. Particularmente, estas redes son utilizadas para encontrar patrones, reconocer objetos, caras y escenas en las imágenes (Calvo, 2017).

Justificación e importancia

Como se mencionó anteriormente, las caídas constituyen uno de los acontecimientos más frecuentes y riesgosos presentados en adultos mayores. Considerando que estas caídas pueden ser detectadas por sistemas de visión artificial de alerta temprana, que permitan una rápida atención que minimice el nivel de riesgo, la implementación de este tipo de sistemas constituye una necesidad social para mejorar las condiciones de pacientes geriátricos.

En la literatura relacionada a la temática se evidencia que mediante el análisis de imágenes es posible recoger las características más importantes para identificar escenas donde se presentan eventos con caídas de personas. Sin embargo, los principales inconvenientes que presentan el uso de estas técnicas, son la dificultad de caracterizar

los fotogramas cuando se presentan cambios de iluminación o de entorno, y, el elevado coste computacional (Belshaw , Taati, Snoek, & Mihailidis, 2011). Por otra parte, los sistemas propuestos requieren de plataformas de hardware y software que son costosas en su implementación y mantenimiento, lo que impide que grupos de bajos recursos económicos puedan adquirir estos sistemas. Conociendo que la mayoría de las personas de la tercera edad tienen reducidos recursos económicos, es necesario proponer un sistema de bajo costo que sea asequible a sus posibilidades y que ayude a mejorar la calidad de vida de estas personas. El propósito de este trabajo de titulación es contribuir en el desarrollo de un sistema de visión artificial que permita solventar las dificultades técnicas, así como las económicas.

Para ello, este proyecto de titulación trabajará en la proposición de un sistema de detección de caídas en adultos mayores que pueda ser implementado en una plataforma de bajo costo. El proyecto considerará el uso de algoritmos que aumenten la eficacia de la detección ante variación de factores ambientales. Adicionalmente, se propondrán algoritmos que puedan ser paralelizados minimizando el coste computacional e implementados en arquitecturas multinúcleo de bajo consumo de potencia.

Alcance del proyecto

A fin de que este proyecto aporte en el desarrollo de un sistema de bajo costo que permita solventar las dificultades técnicas de la detección de caídas mediante inteligencia artificial, se ha planificado que se ejecute los siguientes componentes fundamentales:

1. La instalación y configuración de una plataforma de desarrollo hardware/ software para implementar redes neuronales convolucionales.
2. Selección e implementación del modelo de red neuronal convolucional.
3. Preparación del set de datos para el entrenamiento de la red neuronal.

4. Entrenamiento de la red neuronal convolucional hasta reducir la función de pérdida con un valor cercano a cero.
5. La prueba de concepto del sistema mediante la captura de fotogramas en tiempo real, y con la ayuda de la cámara del computador local.

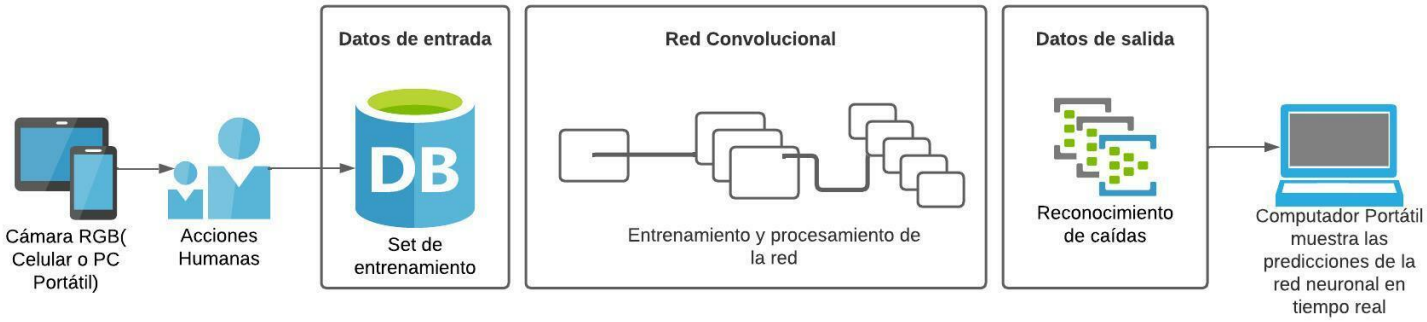
Con la finalidad de desarrollar el primer componente que es la configuración de la plataforma de desarrollo, se ha seleccionado como lenguaje de programación a Python con las librerías asociadas a la inteligencia artificial y aprendizaje profundo. Eso debido a que es una plataforma libre. En el caso de la red neuronal convolucional se va a utilizar del concepto de transferencia de aprendizaje, cuya definición permite utilizar redes neuronales pre – entrenadas. La CNN seleccionada es adaptada para el problema de este trabajo, como es la detección de caídas en personas mayores. Para el entrenamiento de la red neuronal se utilizará varios sets de datos (incluido el set de datos propio) especializados en contener imágenes de personas en diferentes poses como: parado, acostado, caído, agachado y sentado (Adhikari, Kripesh, Bouchachia, & Nait-Charif, 2017) (Casilari & SantoyoRamón, 2018) (Auvinet, Rougier, Meunier, St - Arnaud, & Rousseau, 2010) (Deng, Dong, Socher, Li, & Fei-Fei, 2009). A partir de las cuales se va a determinar cuando la persona se está cayendo. Posteriormente, se procederá a realizar la prueba de concepto del sistema a través de la evaluación de fotogramas en tiempo real para validar si el sistema es capaz de reconocer una caída. Finalmente, en base a las pruebas se realizará ajustes en el sistema para maximizar la eficiencia y exactitud de la detección de caídas del sistema.

De esta manera el diagrama de bloques del sistema de detección de caídas queda configurado como se ilustra en la Figura 1. Una cámara captura las imágenes que son enviada para su análisis hacia la estructura de la red convolucional. Esta red está conformada por la capa de entrada, capas profundas y la capa de salida. Señalando que

entre las capas profundas se encuentran las capas de pooling. Las cuales se encargan de obtener las características más importantes de las imágenes (Liu, Cao, Cui, Song, & Zhao, 2018). En tanto que la capa de salida de la red neuronal está conformada fundamentalmente por las funciones de activación. Quienes se encargan de proporcionar una serie de umbrales que sirven para mapear los datos de salida. En otras palabras, las funciones de activación realizan la comparación de un escenario objetivo de otro. Tarea que es realizada por la función de activación conocida como “Softmax. Por tanto, la capa de salida de la red neuronal identifica cada postura de la persona sea: parada, sentada, acostada, agachada y caída.

Figura 1

Esquema general del sistema propuesto.



Objetivos

Objetivo General

- Diseñar e implementar un sistema de detección de caídas en el adulto mayor mediante visión artificial utilizando redes neuronales convolucionales (deep learning).

Objetivos Específicos

- Seleccionar el entorno de desarrollo integrado IDE, el lenguaje de programación, la base de datos de imágenes de postura humana y las librerías necesarias que permitan desarrollar el sistema de detección de caídas.
- Configurar un sistema de visión artificial utilizando deep learning basado en redes convolucionales.
- Seleccionar e implementar algoritmos para la detección de caídas en adultos mayores.
- Realizar la prueba de concepto del sistema mediante fotogramas en tiempo real

El presente documento resume la ejecución del proyecto de titulación, a fin de facilitar la comprensión de la información se ha dividido en varios capítulos posteriores:

- **Capítulo II: Fundamentos Teóricos y Estado del arte**

  El segundo capítulo detalla las metodologías y técnicas relacionadas a los sistemas de detección de caídas en personas. Mencionando a los sistemas basados en sensores y sus características más relevantes. Y posteriormente, los sistemas basados en inteligencia artificial, que utilizan modelos matemáticos fundamentados en visión artificial y Deep learning.

- **Capítulo III: Desarrollo**

  Este capítulo aborda el diseño y desarrollo del trabajo de titulación. Haciendo un estudio de las técnicas de transferencia de aprendizaje, que permiten la selección de un modelo de CNN pre – entrenado. El objetivo de optar por un modelo pre – entrenado responde a la disminución del tiempo y cantidad de datos, empleado en el entrenamiento de la red neuronal. En cuanto al entrenamiento se mencionan los parámetros que permiten la optimización de recursos computacionales y eficiencia

de la red. Finalmente, se detallan las predicciones realizadas por la red neuronal mediante la cámara del computador local en tiempo real.

- **Capítulo IV: Experimentación y resultados**

  La experimentación y resultados permite evaluar el modelo de CNN, entrenado con el set de datos en particular. Este capítulo aborda el análisis de las métricas de entrenamiento de la red neuronal convolucional como función de pérdida y tasa de aprendizaje. La herramienta Tensorboard se encarga de mostrar gráficamente estas métricas. Finalmente, se muestra la evaluación de la CNN mediante varias predicciones realizadas.

- **Capítulo V: Conclusiones, recomendaciones y trabajo futuro**

  En este capítulo se detallan las conclusiones y recomendaciones al trabajo desarrollado. Adicionalmente, se enlistan los trabajos a futuro que pueden ser realizados tomando como línea base la presente tesis.

- **Anexos**

  En los anexos se incluye el código en Python utilizado para cada una de las actividades que constituyen el trabajo de titulación. Desde la creación del set de datos, el entrenamiento de la CNN, hasta la puesta en marcha del sistema.

Capítulo II. Fundamentos Teóricos y Estado del arte

Introducción

En este capítulo se tratan conceptos base, que permiten conocer las diferentes metodologías y técnicas para el desarrollo de sistemas de detección de caídas en personas. Principalmente, los tipos de sistemas que se detallan son: a) los sistemas de detección de caídas basados en sensores y b) los sistemas basados en inteligencia artificial.

Sistemas de detección de caídas en personas basados en sensores

Para empezar, se va a definir los sensores como dispositivos capacitados para detectar acciones o estímulos externos que permitirán al sistema dar una respuesta. Son elementos encargados de transformar una magnitud física en otra magnitud eléctrica (Corona Ramírez, Abarca Jiménez, & Mares Carreño, 2014). De esta manera se puede apreciar que los sistemas de detección de caídas basados en sensores, establecen su funcionamiento en la recolección de información por medio de la persona y su entorno, para dar una respuesta conforme al análisis de variables eléctricas. Los tipos de sistemas de detección de caídas basados en sensores se pueden clasificar según lo mostrado en la Tabla 2 (Pérolle & Etxeberria Arritxabal, 2017).

Tabla 2

Clasificación de los sistemas de detección de caídas basados en sensores. (Pérolle & Etxeberria Arritxabal, 2017)

	Detección inmediata	Comportamiento inusual
Aparatos portátiles	• Detección de caídas y emisión de alarma inmediata.	• No detectan caídas: Registran un comportamiento

	• Tecnologías utilizadas: sensores de choque, posición, e inclinación y acelerómetros junto con algoritmos de control.	inusual en la persona comparado con un patrón. • No emiten alarmas de forma inmediata. • Tecnologías utilizadas: sensores del ritmo cardíaco, sudoración, etc.
Monitorización ambiental	• Uso de sensores instalados en el ambiente de la persona para detección de una caída. • Tecnologías utilizadas: Utilización de sensores de choque en el piso o alfombras.	• Monitorización de la actividad de la persona. • Tecnologías utilizadas: Sensores de contacto o barreras de infrarrojo (IR) en puertas y ventanas.

Sistemas de detección de caídas en personas basados en inteligencia artificial

El desarrollo de sistemas basados en computadores ha permitido plantear nuevos paradigmas en el área del conocimiento. Uno de ellos es relacionado con la posibilidad de enseñar a las computadoras a realizar tareas que únicamente habían sido encargadas al ser humano. Por ejemplo, tareas como analizar información visual, reconocimiento de voz y objetos, traducción automática en varios idiomas, entre otros (Sanabria S & Archila D, 2011). Esto ha permitido dar origen a una disciplina científica orientada a emular el comportamiento humano, conocida como inteligencia artificial (IA). Esta disciplina se encarga de crear y aplicar algoritmos que permitan a los ordenadores imitar la inteligencia humana (NetApp, 2022). Para este propósito, la inteligencia artificial precisa de los siguientes componentes:

- Sistemas computacionales
- Datos y gestión de los mismos
- Algoritmos de IA avanzados

Con la finalidad de realizar reconocimiento y detección de objetos en imágenes y vídeos es fundamental la tarea de procesamiento de imágenes que es realizado por el sistema de IA. Dentro del procesamiento de imágenes, manejar los conceptos de la visión artificial y el aprendizaje profundo resultan fundamentales en el proceso.

Procesamiento de imágenes

El procesamiento de imágenes es la manipulación de una imagen con fines de mejorarla o extraer información de ella (Brita Inteligencia artificial, 2021). Dentro de los propósitos del procesamiento de imágenes se encuentra el reconocimiento de patrones. Para ello se realizan las siguientes tareas:

- Clasificación: Esta tarea se encarga de clasificar objetos, de forma que al procesar una imagen. El resultado entregado es la clase a la que pertenece el objeto, por ejemplo, perro o gato.
- Detección de objetos: Además de distinguir la clase a la que pertenece el objeto, identifica la posición en X y Y del mismo y dibuja un rectángulo a su alrededor en una imagen o vídeo.
- Segmentación de instancias: Se encarga de dividir en regiones de imagen cada clase de objeto dentro de una escena de imagen o vídeo.

En la figura 2 se ilustra la tarea de clasificación de objetos, donde la salida de la imagen mostrada debe indicar la clase a la que pertenece. En ese caso el objeto pertenece a la clase gato.

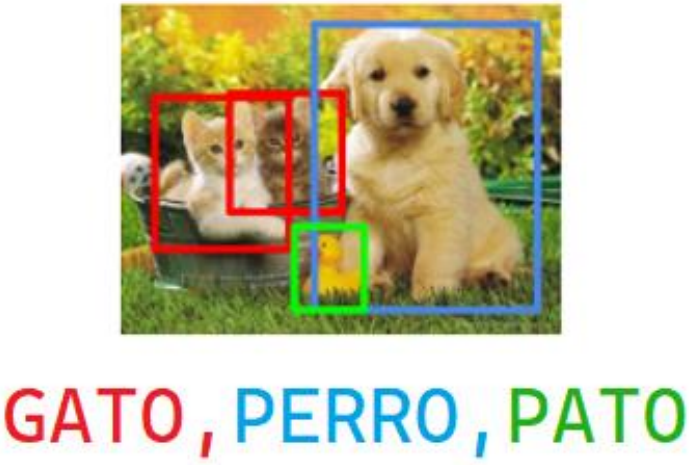
Figura 2

Clasificación de objetos. (Brita Inteligencia artificial, 2021)

La figura 3 muestra la tarea de detección de objetos, mostrando la ubicación y el rectángulo alrededor de cada clase de objeto. Cabe señalar que en una misma escena de imagen o vídeo puede aparecer varias clases de objeto.

Figura 3

Detección de objetos. (Brita Inteligencia artificial, 2021)



En la figura 4 se aprecia la tarea de segmentación de instancias, dividiendo cada clase de objeto en una región de imagen. De la misma forma que en la detección de objetos, en una escena de imagen o vídeo pueden encontrarse varios objetos segmentados.

Figura 4

Segmentación de instancias. (Brita Inteligencia artificial, 2021)

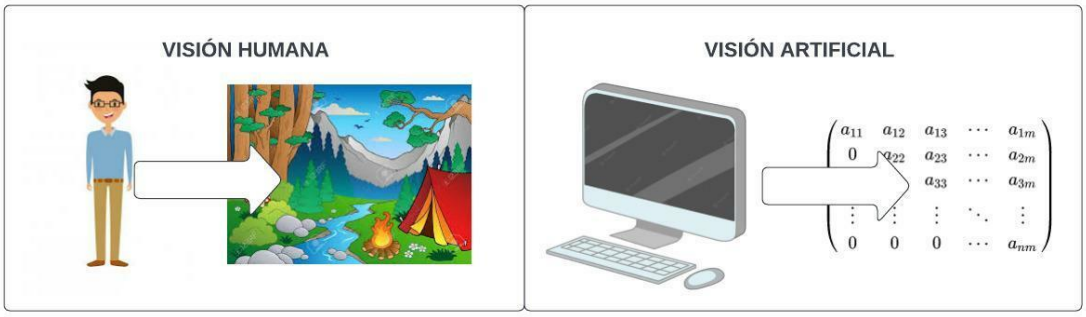
Visión artificial

La visión artificial es un campo de la IA que se centra en el desarrollo de algoritmos y técnicas, que permiten que las computadoras y otros sistemas adquieran información significativa de imágenes y videos. Por tanto, si la inteligencia artificial hace que las computadoras piensen, la visión artificial les permite ver y observar de una forma similar al ojo humano. Por esta razón, para entender de mejor forma que es la visión artificial, se habla primeramente de la visión humana. La misma que se define como la capacidad de interpretar un entorno mediante los rayos de luz que llegan al sentido de la vista. En otras palabras, este proceso radica en la entrada de luz hacia el ojo humano mediante la córnea. Seguidamente, la córnea dirige la luz a las pupilas y el iris, que se encargan de controlar la cantidad de luz que ingresa al ojo. Concluyendo que la visión humana es una tarea de procesamiento de información, ya que nuestro cerebro es capaz de procesar esta información de entrada, en características palpables como: color, forma, movimiento, etc. (Gamboa Uribe, 2020).

En el caso de la visión artificial, es el computador con sus periféricos el que va a observar el entorno. Una imagen es interpretada por el computador como una matriz compuesta por un número finito de elementos. Cada elemento(pixel) ocupa una posición en el plano cartesiano con coordenadas x,y, y un valor asociado al color de la imagen en dicha posición. De aquí se señala al píxel como la unidad mínima de medida de una imagen. En la figura 5 se puede apreciar como la visión humana y artificial perciben el entorno o ambiente.

Figura 5

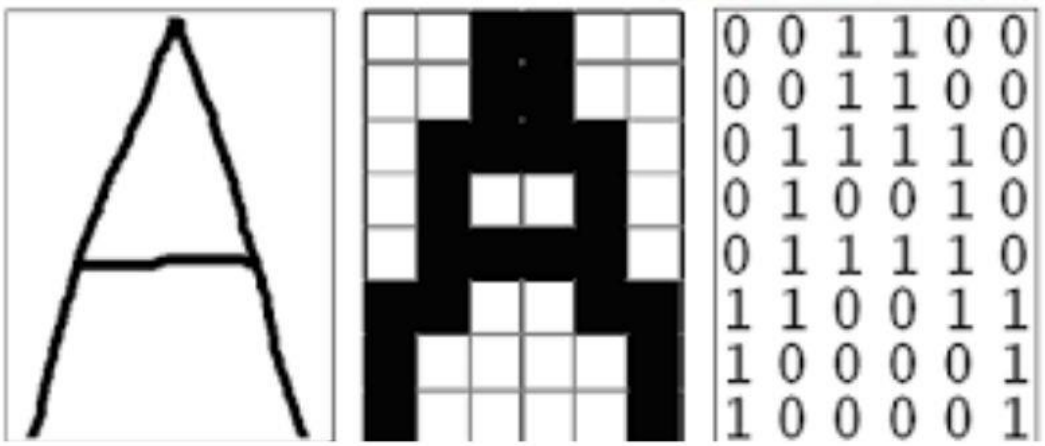
Visión humana vs visión artificial.



En lo que respecta al color de una imagen, se considera la imagen a blanco/negro y color. En el caso de la imagen blanco/negro, la visión artificial lo interpreta con una representación binaria. Es decir, se asignan valores de 0 y 1 al blanco y negro respectivamente, como se observa en la figura 6 (Gamboa Uribe, 2020).

Figura 6

Visión artificial y su interpretación de la imagen blanco/negro. (Gamboa Uribe, 2020)

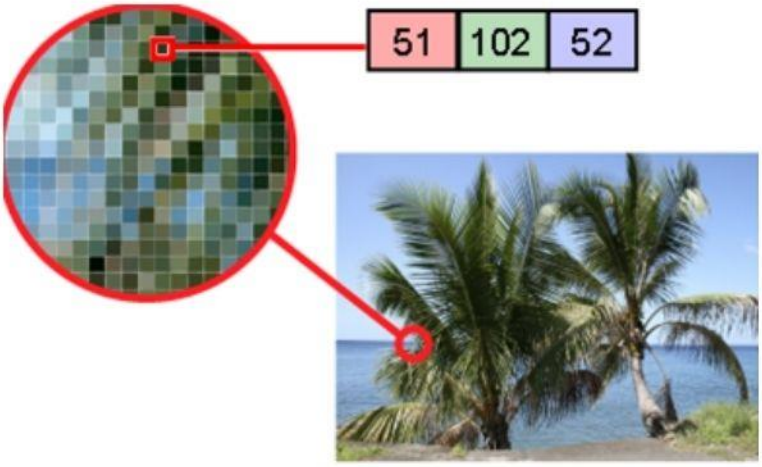


Por otro lado, en la imagen a color la representación viene dada por tres valores correspondientes a la cantidad(tonos) de rojo, verde y azul, conocido en inglés como Red,

Green, Blue (RGB). En consecuencia, cada píxel (o punto) de la imagen a color tiene asignado tonos de rojo, verde y azul que definen el color del pixel. Por cada color se requiere de 1 Byte de información es decir permite valores comprendidos entre el 0 al 255, siendo 0 ausencia del color y 255 el valor máximo. Así por ejemplo el color amarillo tono fuerte será codificado en RGB como (255,255,0) y el color blanco como (255,255,255). La figura 7 ejemplifica un tono de verde.

Figura 7

Visión artificial y su interpretación de la imagen a color. (Gamboa Uribe, 2020)



Etapas de un sistema de visión artificial

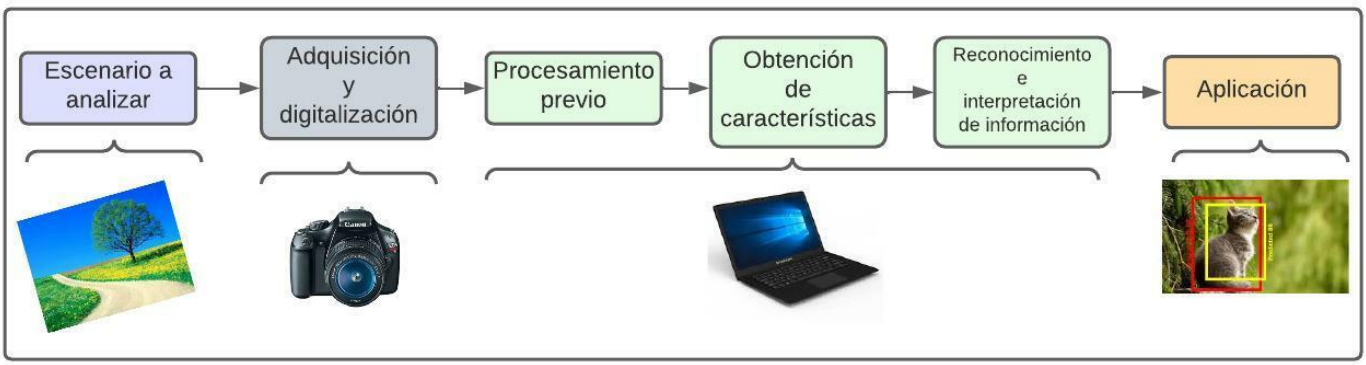
Los sistemas de visión artificial pueden variar según la aplicación para la que se hayan diseñado (Solución Ingenieril, 2022). No obstante, en forma generalizada estos sistemas se componen de las siguientes etapas:

- **Escenario a analizar:** Es el área donde se encuentra la información que se requiere analizar.
- **Adquisición y digitalización:** Es el proceso de captación del escenario a analizar y posteriormente dar un formato digital, para que pueda ser procesado por las etapas siguientes.

- **Procesamiento previo:** Incluye las acciones y técnicas asociadas a reducir el ruido de la imagen.
- **Obtención de características:** Esta etapa es muy importante, ya que aquí se realiza el proceso de extracción de características que son de interés. Las acciones que se realizan aquí son: detección de bordes, esquinas, colores y realce de objetos de importancia.
- **Reconocimiento e interpretación de información:** Proceso de interpretación de las imágenes, dando un significado al conjunto de objetos reconocidos para que puedan ser tratados por la etapa de aplicación.
- **Aplicación:** Esta etapa se encarga de dar solución a un problema específico mediante la información que se procesó. Las aplicaciones de visión artificial son detalladas en la siguiente sección.

Figura 8

Etapas de un sistema de visión artificial. (Solución Ingenieril, 2022).



Aplicaciones de la visión artificial

El desarrollo de esta disciplina ha permitido su aplicación en diferentes sectores como el industrial, automoción, seguridad e incluso en la medicina. Donde la gestión y

análisis de la información es fundamental para su diario funcionamiento (Sanabria S & Archila D, 2011). Algunas de las áreas donde se aplica visión artificial son:

- **Medicina:** El análisis e interpretación de imágenes por parte de los ordenadores, permite la detección de patologías, enfermedades y tumores (Castrillón Fernández, Bolaños, Enríquez, & Pencue-Fierro, 2007).
- **Vigilancia/Seguridad:** Por medio de algoritmos de visión artificial se puede detectar la presencia de personas extrañas en cierto espacio o lugar (Jiménez, 2009).
- **Industria:** En este ámbito la visión artificial está directamente relacionada a mejorar el control de calidad de los procesos industriales y en consecuencia obtener mejores productos (Medina Muñoz, González-López, Mayorquín Robles, & López Valencia, 2019), (Ministerio de Educación, 2012).
- **Robótica:** La visión artificial encaja muy bien en este campo, ya que brinda mayor autonomía a los robots, permitiendo desarrollar actividades de carácter logístico o industrial. Algunas de estas actividades van desde el reconocimiento de objetos con formas complejas, hasta guiar al robot a una ubicación de interés en específico (Pelegrí, 2019).
- **Interacción Humano – computadora:** Las personas con capacidades limitadas se ven beneficiadas con el desarrollo de métodos, que les permita interactuar con la computadora, sin necesidad de los dispositivos habituales teclado – ratón. Esta interacción se lleva a cabo con la detección y movimiento del rostro de la persona. Y comprende entre algunos aspectos, el movimiento de la cabeza y el guiño de cada ojo (Osimani & Kohmann, 2015).

Un sistema con inteligencia artificial puede sensar, razonar y adaptarse de acuerdo al escenario. Para ello se puede utilizar algoritmos de machine learning para enseñarle cosas o se puede utilizar un conjunto de algoritmos específicos que permiten que el sistema aprenda por sí solo, de manera similar al cerebro humano, lo que se conoce como Deep learning. La detección de caídas en adultos mayores realizada en este trabajo está precisamente basada en Deep Learning.

Deep Learning

El Deep learning o aprendizaje profundo es una técnica de vanguardia utilizada en la inteligencia artificial. Mediante esta técnica en lugar de enseñar a los computadores como es un objeto para detectarlo, se le entrega suficiente información del objeto para la máquina pueda resolverlo por sí mismo. Por ejemplo, si se quiere detectar por visión artificial un ratón, en lugar de enseñarle como es el ratón se le da suficientes fotos de ratones y la máquina resolverá por si sola si es un ratón o no. Actualmente el Deep learning se utiliza en la detección de objetos, reconocimiento de voz, conducción de autos autónomos, o en la búsqueda de células cancerígenas. El modelo de Deep Learning se fundamente en el uso de las redes neuronales artificiales (Pérez Lorenzo, 2019).

Redes Neuronales Artificiales (ANN)

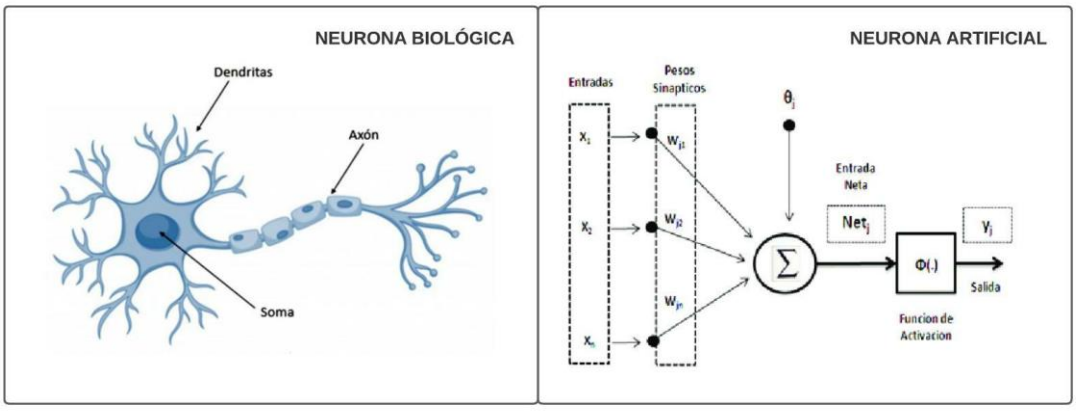
Las redes neuronales artificiales o en inglés artificial neural network (ANN), están basadas en el funcionamiento de las redes neuronales biológicas del cerebro humano. Las neuronas de nuestro cerebro se componen de tres elementos: dendritas, el soma y el axón (Pose, 2009). Las dendritas se encargan de la recepción de estímulos externos provenientes de otras neuronas. El soma es el espacio donde se procesan estos estímulos y es el encargado de generar la mayor parte de las moléculas de la neurona. El axón transmite los impulsos nerviosos recibidos desde el soma hacia las neuronas contiguas. En resumen, las dendritas de una neurona recogen las señales de otras

neuronas, que son procesadas en el soma y enviadas a lo largo del axón a otras neuronas (Olabe, 1998).

En contraparte, las neuronas artificiales se constituyen por los pesos de las entradas, bias y función de activación. Haciendo analogía a los elementos de la neurona biológica del cerebro humano, los pesos de entrada representan a las dendritas ya que recogen información de las entradas para que sea procesada en el bias (análogo al soma). El bias realiza la suma de todos los pesos y es estimulado por una función de activación (paralela a la función del axón) para llevar la información a la salida de la neurona artificial. En la figura 9 se aprecia los dos tipos de neurona biológica y artificial, con sus principales elementos.

Figura 9

Neurona Biológica vs Neurona Artificial. (Xeridia, 2019), (Gámez Albán, 2016).

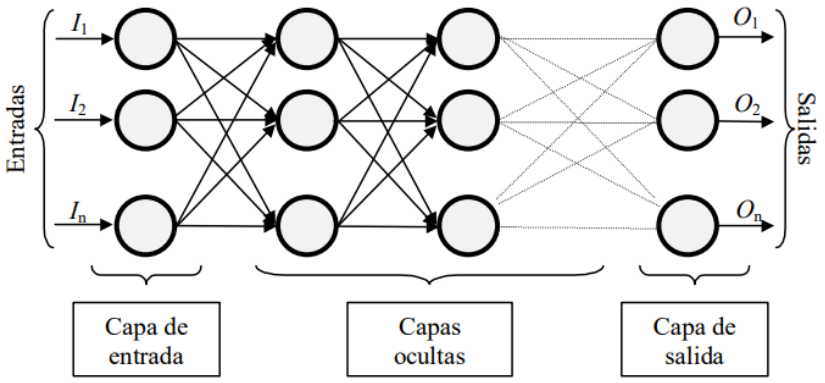


La arquitectura de las redes neuronales artificiales se encuentra organizada por capas. La capa de entrada recibe la información mediante un conjunto de datos estructurado. La capa oculta está formada por varias capas, quienes cumplen determinadas acciones de extracción de características. Y la capa de salida muestra la información procesada conforme a la aplicación desarrollada, sea clasificación de imágenes, detección de objetos, reconocimiento de voz, etc. El esquema de una red

neuronal artificial se muestra en la figura 10 (Flórez López & Férnandez Férnandez, 2008).

Figura 10

Esquema de una Red Neuronal Artificial. (Matich, 2001).



De entre los tipos de redes neuronales artificiales, las redes neuronales convolucionales (CNN) son aquellas que han logrado mejores resultados en tareas de procesamiento de imágenes. Varios trabajos (Núñez-Marcos, Azkune, & Arganda-Carreras, 2017), (Sherin P, 2019) hacen uso de modelos de CNN existentes como: AlexNet, VGG-16, ResNet, Inception v3, para tareas de visión artificial. Los cuales demuestran mayor precisión y eficiencia que los modelos primitivos de redes neuronales artificiales. Estos modelos de CNN se caracterizan por tener varias capas profundas, en donde se realiza el proceso de extracción de características. Teniendo como actor principal la operación de convolución, que se hablara más adelante.

Redes neuronales convolucionales (CNN)

Una red neuronal convolucional o en inglés convolutional neural network (CNN), es un tipo de red neuronal que emplea operaciones de convolución para el procesamiento de imágenes. Siendo importante señalar que, las CNN representan uno de los principales algoritmos en el desarrollo y perfeccionamiento de la visión por computadora (Calvo,

2017). La convolución es una herramienta matemática muy útil en el procesamiento de imágenes, debido a que mediante el uso de filtros convolucionales se pueden obtener efectos con características especiales de una imagen.

La convolución es un operador que transforma dos funciones f y g en una tercera función. Dicha función representa la magnitud en la que se superponen f y una traslación invertida de g. La representación del operador convolucional es con el símbolo $*$, y está definido por la siguiente ecuación matemática:

$$f(t) * g(t) = \int_0^t f(u)g(t-u)du; \quad \forall t > 0$$

Para funciones discretas se usa la forma discreta de convolución:

$$x[n] * y[n] = \sum_{k=0}^{n} x_k \; y_{n-k}$$

Como se puede apreciar de las ecuaciones anteriores, la convolución se encuentra definida tanto para funciones continuas como discretas. Siendo la integral y serie respectivamente para cada clase (Percat Fedalto, 2015).

El filtro convolucional o kernel está definido como una función bidimensional $z = F(x, y)$ donde *x,y* son coordenadas espaciales, y *z* representa el valor de la intensidad de la imagen en el punto (x, y) (Giménez-Palomare, Monsoriu, & Alemany-Martínez, 2016). Tanto la imagen como el filtro convolucional son consideradas como funciones bidimensionales, de manera que la ecuación de convolución queda definida por la siguiente ecuación, en donde, la función $f(x, y)$ representa la imagen y la función $g(x, y)$ es el filtro o kernel que se aplicará (Oppenhein & Schafer, 2010):

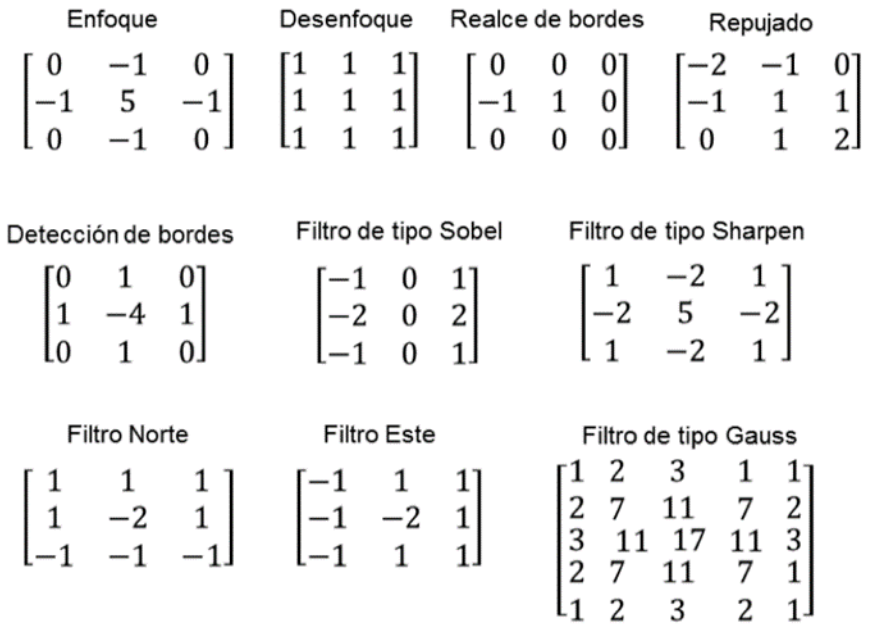
$$f(x,y) * g(x,y) = \iint_{-\infty}^{\infty} f(\alpha,\beta)g(x-\alpha, y-\beta)d\alpha d\beta$$

Donde,

- *x* y *y,* son las coordenadas del pixel.
- α y β son variables utilizadas para el desplazamiento de la matriz de convolución.

Figura 11

Filtros o kernel más usados. (Giménez-Palomare, Monsoriu, & Alemany-Martínez, 2016).



En cuanto al filtro, habitualmente se usan matrices de orden $3x3$ o $5x5$. En la figura 11 se ilustran algunos tipos de filtros más usados de acuerdo a la característica que se quiera obtener. Las figuras 12, 13, 14, 15 y 16 muestran el resultado de una imagen al aplicarse diferentes tipos de filtros como de enfoque, de desenfoque, de repujado, de

detección de bordes y de Sobel respectivamente (Giménez-Palomare, Monsoriu, & Alemany-Martínez, 2016).

Figura 12

Filtro Enfoque

0	-1	0
-1	5	-1
0	-1	0

Figura 13

Filtro Desenfoque

1	1	1
1	1	1
1	1	1

Figura 14

Filtro de repujado

-2	-1	0
-1	1	1
0	1	2

Figura 15

Detección de bordes

0	1	0
1	-4	1
0	1	0

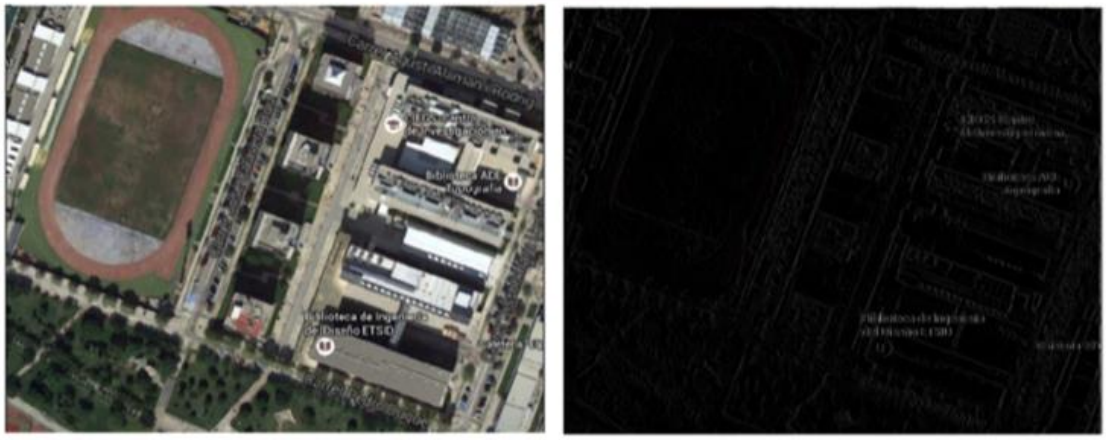
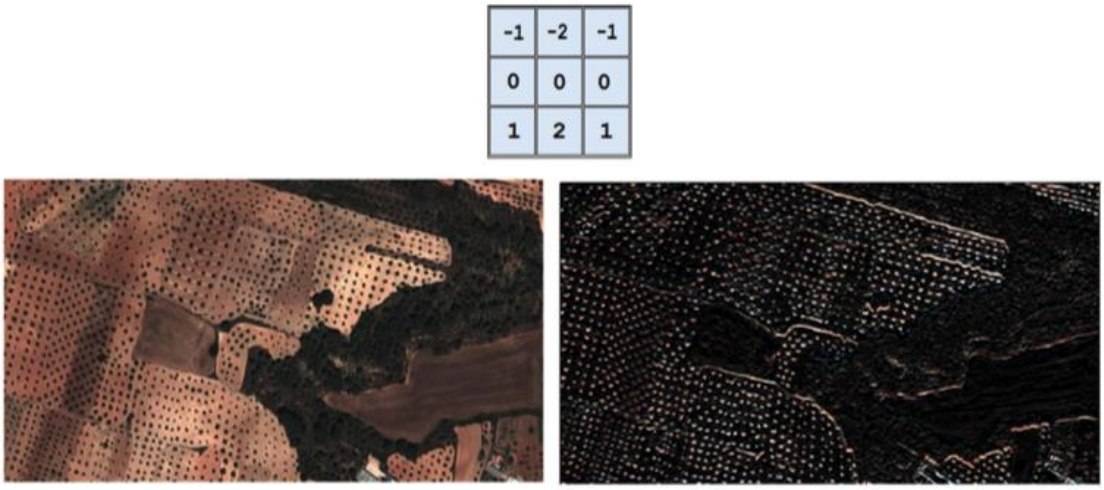


Figura 16

Filtro de Sobel



Arquitectura de las CNN

Una CNN para ejecutar algoritmos de Deep Learning está constituida por varias capas ocultas. Cada capa oculta tiene como objetivo disminuir la carga computacional y detectar líneas, curvas o bordes para realizar tareas como: el reconocimiento de objetos, personas, etc.

Si bien es cierto, las CNN están constituidas por varias capas. Según, la función que cumplen se pueden identificar tres tipos de capas (Pérez Lorenzo, 2019):

- Capas Convolucionales
- Capas de Pooling
- Capas totalmente conectadas (Fully Connected)

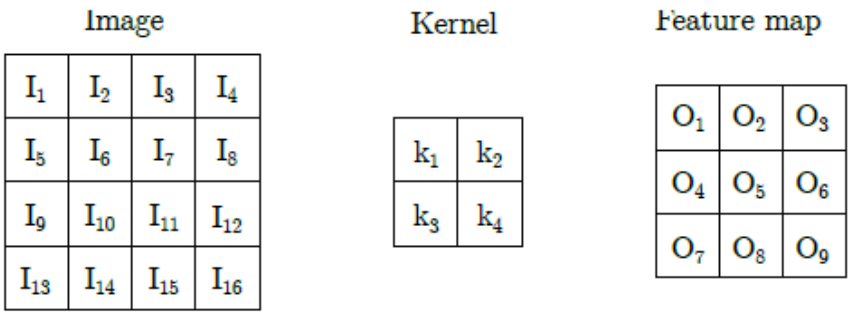
La capa convolucional es el núcleo de la arquitectura de una CNN. En esta capa se aplica la operación de convolución (filtros o kernel) a la imagen de entrada. El resultado se almacena en una matriz conocida como mapa de características o matriz de activación, que se caracteriza por ser de menor tamaño que la imagen original. Este resultado

intermedio actuará como entrada para la siguiente capa y siendo la imagen de menor tamaño que la original reducirá el coste computacional para la siguiente capa.

El proceso de convolución entre la matriz que representa la imagen y kernel inicia cuando la kernel recorre cada píxel de la matriz de la imagen. El filtro recorre de izquierda a derecha y de arriba hacia abajo, cada uno de los pixeles del área de acción, según sea el orden de su matriz $3x3$ o $5x5$. La figura 17 muestra el tamaño de las matrices que intervienen en el proceso de convolución y su resultado (mapa de características).

Figura 17

Matriz de la imagen, kernel y activación. (Arriola Oregui, 2018).



Para la obtención de cada elemento del mapa de características o matriz de activación se recorre el filtro (o kernel) sobre la imagen de entrada, como se indica en la figura 18. Particularmente para encontrar los tres primeros elementos del mapa de características, se realizan las siguientes operaciones:

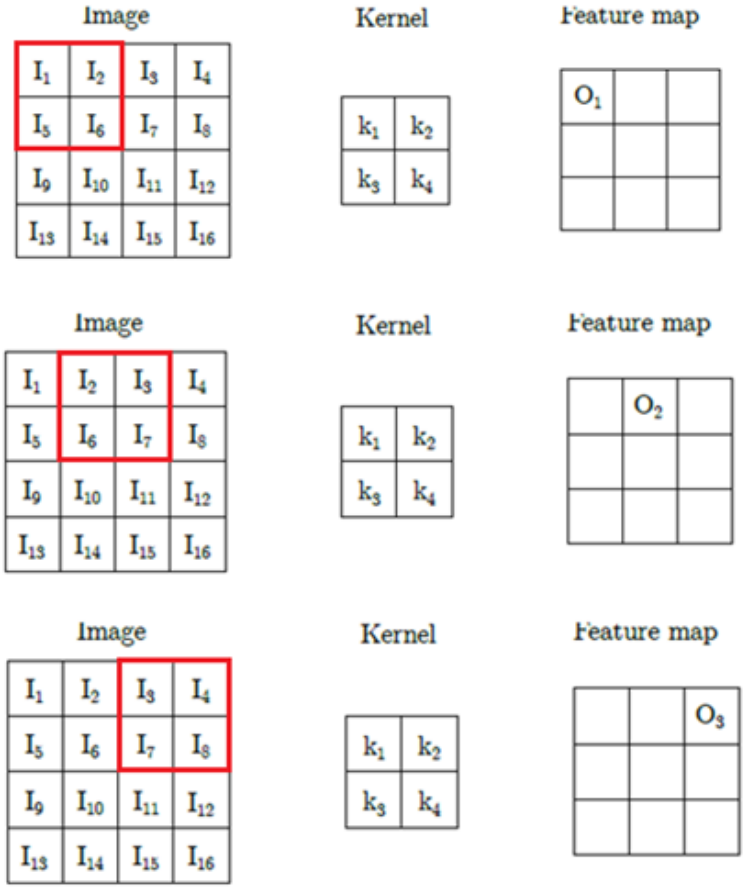
$$(I_1.k_1) + (I_2.k_2) + (I_5.k_3) + (I_6.k_4) = \boldsymbol{O_1}$$

$$(I_2.k_1) + (I_3.k_2) + (I_6.k_3) + (I_7.k_4) = \boldsymbol{O_2}$$

$$(I_3.k_1) + (I_4.k_2) + (I_7.k_3) + (I_8.k_4) = \boldsymbol{O_3}$$

Figura 18

Proceso de convolución. (Arriola Oregui, 2018)



Finalmente, se debe mencionar los parámetros que rigen el comportamiento adecuado de la capa convolucional (de López Diz, 2019), (Vázquez Rull, 2016):

1. **Tamaño del kernel:** Define el tamaño del filtro aplicado a la imagen de entrada con la operación de convolución. Se corresponde con las características que se desea extraer de la entrada.
2. **Stride:** Corresponde al desplazamiento que realiza el filtro sobre la imagen de entrada en cada iteración. El incremento de este parámetro provoca volúmenes de salida parcialmente más pequeños.

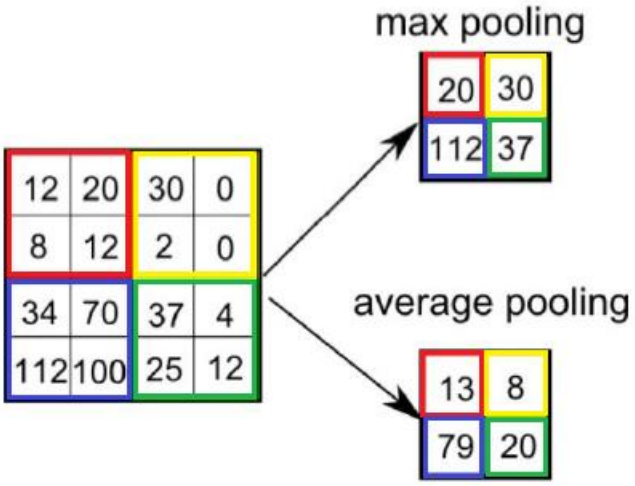
3. **Padding:** Consiste en la aplicación de operaciones (incluir píxeles adicionales) que evitan la desaparición del mapa de características. Debido a que el tamaño de la matriz de salida es siempre menor que la imagen original.

La capa de Pooling cumple con la función de reducir la dimensionalidad, así como los parámetros de la CNN. En consecuencia, los resultados de esta capa permiten disminuir el tiempo de entrenamiento y coste computacional de la red.

Existen fundamentalmente dos tipos de capas de Pooling que resumen estadísticamente los valores de salida de una región. Entre las que destacan: *Average Pooling,* calcula el valor medio (o promedio) entre los valores definidos en función del tamaño del filtro. Y el *Max Pooling,* que obtiene el valor máximo de los valores definidos en una región (Pérez Lorenzo, 2019).En la figura 19 se muestra el Average y Max Pooling determinando el valor máximo y promedio de cada región respectivamente.

Figura 19

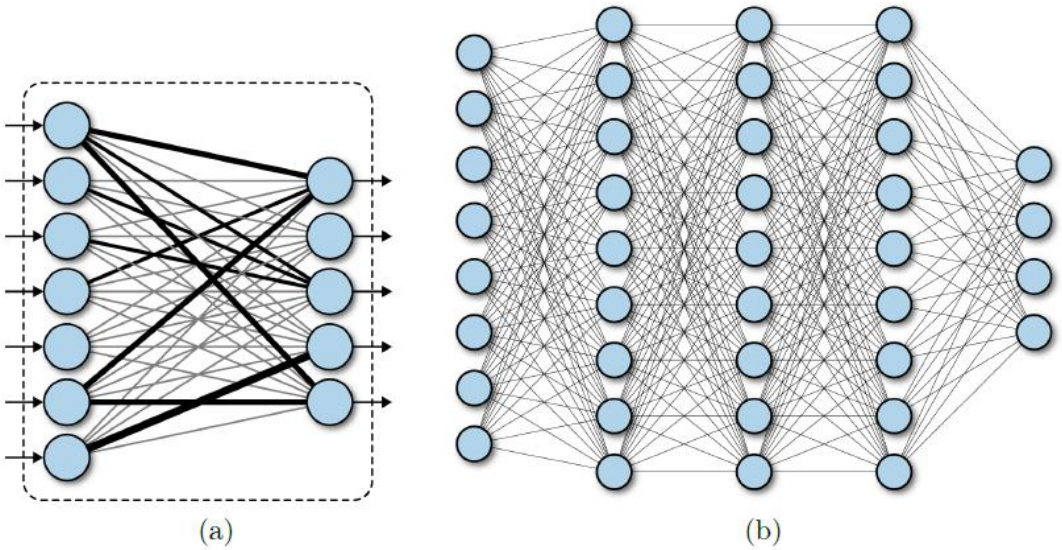
Operaciones de Max Pooling y Average Pooling. (Pérez Lorenzo, 2019)



La capa Fully Connected se caracteriza por tener conectadas absolutamente todas sus neuronas con la capa anterior. Además, esta capa se aplica al final de la arquitectura de una CNN. Con el propósito de pasar la información a una capa de clasificación (softmax), que determina a que clase pertenece la imagen de entrada. La capa Fully Connected puede ser de una sola capa o multicapa como se muestra en la figura 20, donde a) capa Fully Connected b) Multicapa Fully Connected.

Figura 20

Capa Fully Connected. (Pérez Lorenzo, 2019).



Funciones de activación de las CNN

Las funciones de activación aportan con factores no lineales en las CNNs. De modo que, la red neuronal puede aproximarse arbitrariamente a cualquier modelo no lineal y más complejo. Además, proporcionan una serie de umbrales que son útiles para el procesamiento de los datos de salida. Entre las funciones de activación más utilizadas están: lineal, unidad lineal rectificada o ReLU y softmax

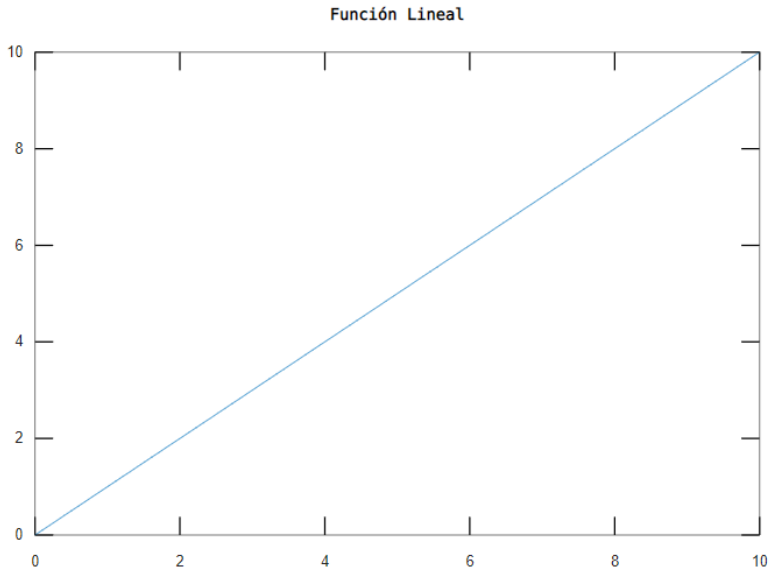
Función Lineal

Esta función también es conocida como identidad, ya que permite que la entrada sea igual que la salida. Es utilizada en problemas donde la salida requiere de una regresión lineal y por tanto sea un valor único. La representación matemática de esta función es:

$$f(x) = x$$

Figura 21

Representación Función Lineal.



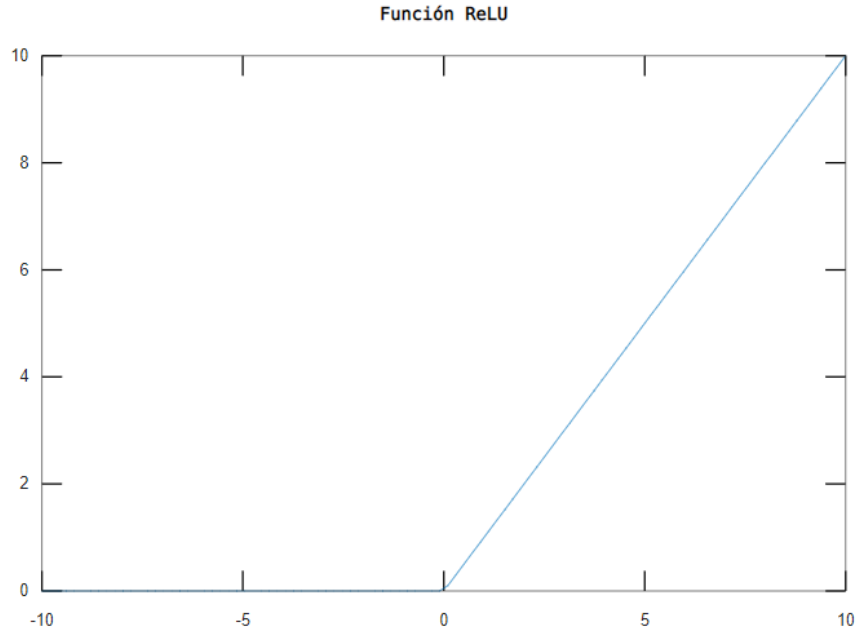
Unidad Lineal Rectificada o ReLU

La función de activación ReLU es una de las más utilizadas en el diseño de arquitecturas de redes neuronales profundas (Glorot, Bordes, & Bengio, 2011). Lo que hace la función ReLU es cambiar la salida por cero si los datos de entrada son valores negativos y conservar esa salida si los datos de entrada son valores positivos. La representación matemática de esta función es:

$$ReLU(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$

Figura 22

Representación Función ReLU.



Función Softmax

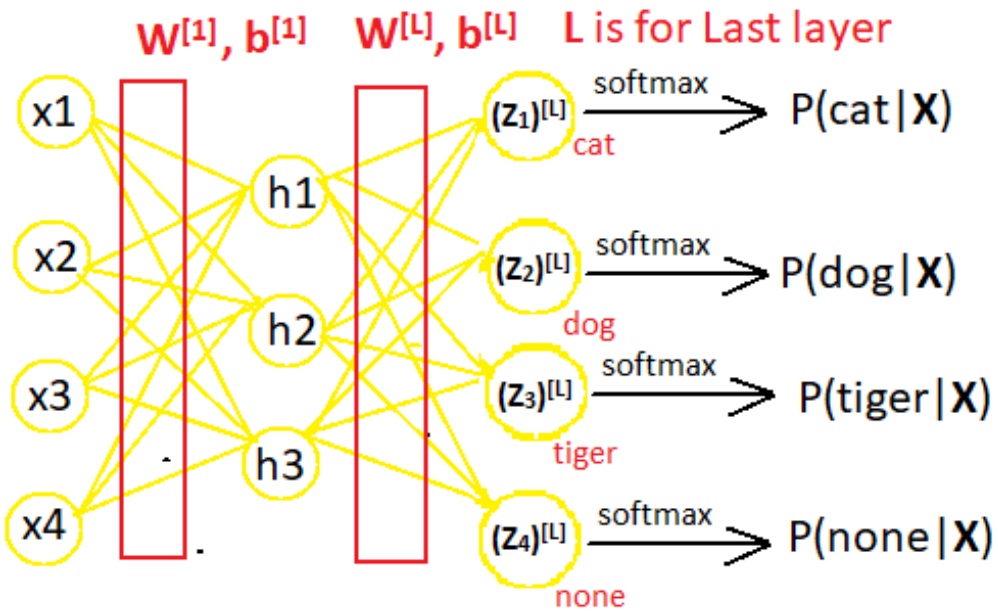
La función Softmax (o Regresión logística multinomial) es una extensión de la función logística, empleada para clasificación multiclase. Esta función convierte un vector de números en un vector de probabilidades. Donde la salida es una distribución de probabilidades pertenecientes a cada clase de la clasificación (Zúñiga-López, Avilés-Cruz, Ferreyra-Ramírez, & Rodríguez-Martínez, 2020). La representación matemática de esta función es:

$$Softmax\,(z_i) = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}$$

La salida de la función Softmax como resultado de una distribución de probabilidad muestra valores comprendidos entre 1 y 0. Siendo los valores más cercanos a 1, los que determinen el trabajo de la red neuronal convolucional.

Figura 23

Función Softmax. (Coder Solution, 2022).



Set de datos de entrenamiento de las CNN

El éxito del funcionamiento del sistema depende del entrenamiento de las redes neuronales convolucionales. De allí la gran importancia que se da al proceso de selección y gestión del conjunto de datos que servirán para capacitar la CNN y que la red aprenda de los ejemplos para cumplir con su propósito. En el caso particular de este proyecto, se debe utilizar set de datos para la detección de caídas en personas. Por ejemplo, el conjunto de datos de UR Fall Detection Dataset y Fall detection Dataset contienen secuencias de imágenes de varias personas. Estas secuencias de imagen están conformadas por actividades de la vida diaria y caídas.

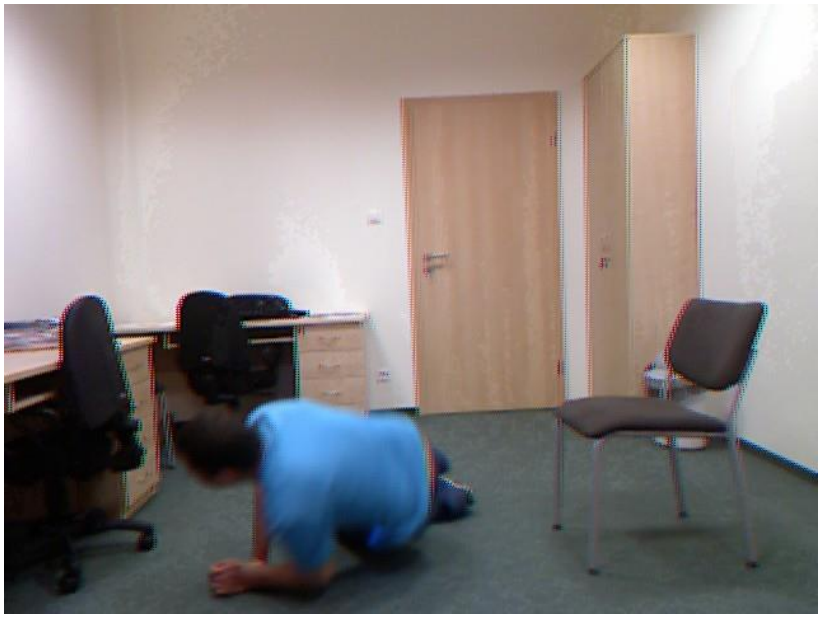
En UR Fall Detection Dataset (Kępski & Kwolek, 2014) se encuentran 70 secuencias de imagen, distribuidas entre 30 para caída y 40 para actividades diarias. En la figura 24 y 25 se aprecia dos de estas secuencias.

Figura 24

Secuencia de actividad diaria en UR Fall Detection Dataset. (Kępski & Kwolek, 2014)

Figura 25

Secuencia de caída en UR Fall Detection Dataset. (Kępski & Kwolek, 2014).



En Fall Detection Dataset (Adhikari, Kripesh, Bouchachia, & Nait-Charif, 2017) de forma similar al anterior dataset, se encuentran secuencias de imágenes correspondientes a caídas y actividades diarias de cinco personas diferentes. Que están clasificadas en cinco categorías de poses: de pie, sentado, acostado, agachado y gatear. En la figura 26 y 27 se presenta dos secuencias diferentes de este conjunto de datos.

Figura 26

Secuencia de actividad diaria en Fall Detection Dataset. (Adhikari, Kripesh, Bouchachia, & Nait-Charif, 2017)

Figura 27

Secuencia de caída en Fall Detection Dataset. (Adhikari, Kripesh, Bouchachia, & Nait-Charif, 2017)

Capítulo III. Desarrollo

Introducción

Una vez revisado los fundamentos teóricos y estado del arte, en este capítulo se abordará el diseño y desarrollo de este proyecto. Empezando por la instalación de la plataforma de hardware y software, que permiten la programación y entrenamiento de la red neuronal convolucional. Posteriormente, se detallan los conceptos base de la transferencia de aprendizaje para la selección de un modelo de CNN pre – entrenado, que disminuya los tiempos de entrenamiento y cantidad de datos empleado. Finalmente, se detallará la creación de una página web mediante la que se realiza la detección de caídas en personas, en tiempo real.

Plataforma de desarrollo de hardware y software

El hardware más importante para el desarrollo de este trabajo es la Unidad de Procesamiento Gráfico (GPU). Ya que, él mismo permite optimizar la fase de entrenamiento de la CNN, mediante el cálculo de operaciones de convolución en matrices de gran dimensión.

El conjunto de programas o software utilizada en este proyecto está basado en el lenguaje de programación Python, librerías, y la Interfaz de programación de aplicaciones (API) de detección de objetos. En la tabla 3 y 4 se detalla el hardware y software usados respectivamente.

Tabla 3

Especificaciones de Hardware

Hardware	Características y especificaciones
Computador Portátil HP Pavilion Gaming	• AMD Ryzen 7 4800H con Radeon Graphics 2.90 GHz.

	• RAM 12 GB. • Windows 11 Home Single Language
Unidad de Procesamiento Gráfico (GPU) NVIDIA GeForce GTX 1650 Ti	• Memoria total disponible para gráficos 9914 MB • Memoria de vídeo dedicada 4096 MB GDDR6
Cámara web integrada en computador	• Integrada en el computador

Tabla 4

Especificaciones de Software

Software	Características y especificaciones
LabelImg	Permite el etiquetado de imágenes, de las diferentes clases a utilizar en el entrenamiento de la CNN y conversión a archivos xml.
Anaconda	Es una suite de código abierto que contiene una serie de aplicaciones, librerías y entornos diseñados para el desarrollo de programas en Python.
Jupyter notebook	Es un entorno de desarrollo de código abierto que permite crear, compartir y ejecutar código.
API de detección de objetos de TensorFlow	Es una herramienta que permite configurar modelos pre – entrenados de

	redes neuronales profundas, para crear sus propios clasificadores y detectores de objetos.

Instalación y configuración del software utilizado

Instalación de Anaconda, Cuda y CuDNN

La instalación empieza con Anaconda, una herramienta que permite administrar paquetes, librerías y configurar entornos virtuales de programación. Anaconda es el entorno que permite instalar desde el lenguaje de programación a utilizar (Python), así como las librerías utilizadas en Deep Learning. Dentro de estas librerías se encuentra TensorFlow, cv2, NumPy y las librerías de detección de objetos. En la tabla 5 se presenta una breve descripción de las librerías de Deep learning usadas.

Figura 28

Fuente (Anaconda Inc, 2022)

Tabla 5

Librerías utilizadas de Deep Learning

Librería	Descripción
TensorFlow	Es una librería de código abierto utilizada para el cálculo numérico, en aprendizaje automático.
Cv2	OpenCV es una librería de código abierto para visión artificial, que permite la detección de movimiento, reconocimiento de objetos, entre otras aplicaciones.
NumPy	Biblioteca de matemática para trabajar con arreglos de n – dimensiones.
Nvidia CuDNN	Es una biblioteca de aceleración de GPU para redes neuronales profundas. Se encarga de reducir la sobrecarga de memoria y mejorar el rendimiento.

Dado que para mejorar el desempeño en el entrenamiento de la red neuronal convolucional se va a utilizar una GPU Nvidia, es necesario trabaja con CUDA. CUDA es una plataforma de computación en paralelo que incluye un compilador y conjunto de herramientas de desarrollo, que mejoran significativamente los cálculos matemáticos, asociados al entrenamiento de la CNN. La descarga de CUDA se realiza desde la página oficial de Nvidia (Nvidia Developer, 2022).

Una vez descargado e instalado CUDA, se procede agregar la librería CuDNN a la plataforma de desarrollo. CuDNN se encarga de acelerar y mejorar el rendimiento de

la GPU. Para la instalación de la librería, simplemente se copian los archivos descargados de la carpeta CuDNN en la carpeta de instalación de CUDA. La descarga de CuDNN se realiza desde la página oficial de Nvidia (Nvidia Developer, 2022).

Instalación de la API de detección de objetos de TensorFlow

La API de detección de objetos de TensorFlow permite adaptar a este proyecto, modelos de redes neuronales convolucionales previamente entrenados. Para su instalación se siguen los siguientes pasos:

1. Se crea y asigna una nueva carpeta con el nombre TensorFlow (Por ejemplo: C:\Users\TensorFlow).
2. Se descarga los archivos de configuración de la API de detección de objetos, desde el repositorio de TensorFlow Model Garden (Yu, y otros, 2020) y colocarlos dentro de la carpeta creada.
3. Dado que la API de detección de objetos de TensorFlow utiliza protobuf, el siguiente paso es la descarga de Protobuf. Protobuf es un formato de intercambio de datos que permite almacenar e intercambiar datos estructurados. La descarga de Protobuf se realiza desde (GitHub, 2022).
4. Luego, en una terminal dentro del directorio TensorFlow/models/research se ejecuta el comando correspondiente para la instalación de Protobuf, como se puede observar en la figura 29.

Figura 29

Comando de instalación de Protobuf. (Vladimirov, 2022).

```
# From within TensorFlow/models/research/
protoc object_detection/protos/*.proto --python_out=.
```

Instalación de LabelImg para el etiquetado de imágenes

LabelImg es un programa que permite crear regiones de interés dentro de una imagen. Estas regiones de interés se llevan a cabo mediante etiquetas sobre los objetos a detectar. Para la instalación de LabelImg se siguen los siguientes pasos:

1. Descarga del paquete de LabelImg, desde su repositorio (Tzutalin, 2015).
2. De preferencia incluir la carpeta de LabelImg dentro de la carpeta de TensorFlow, creada en la sección anterior.
3. Abrir una ventana de terminal y ejecutar el comando que se indica en la figura 30.

Figura 30

Comando de instalación de LabelImg. (Vladimirov, 2022).

```
conda install pyqt=5
pyrcc5 -o libs/resources.py resources.qrc
```

4. Para ejecutar el programa se abre una nueva terminal desde el directorio donde se encuentra la carpeta de LabelImg. Y se ejecuta cualesquiera de los dos comandos mostrados en la figura 31.

Figura 31

Comando para ejecutar LabelImg. (Vladimirov, 2022).

```
python labelImg.py
# or
python labelImg.py [IMAGE_PATH] [PRE-DEFINED CLASS FILE]
```

Finalmente, la tabla 6 muestra las versiones de software instaladas, con las que se diseña y desarrolla el sistema de detección de caídas en personas. Cabe mencionar que antes de su instalación, se ha verificado la compatibilidad tanto con el sistema

operativo del computador local y de los programas entre sí. Con el objetivo de prevenir errores informáticos a futuro.

Tabla 6

Versiones de software y librerías instaladas

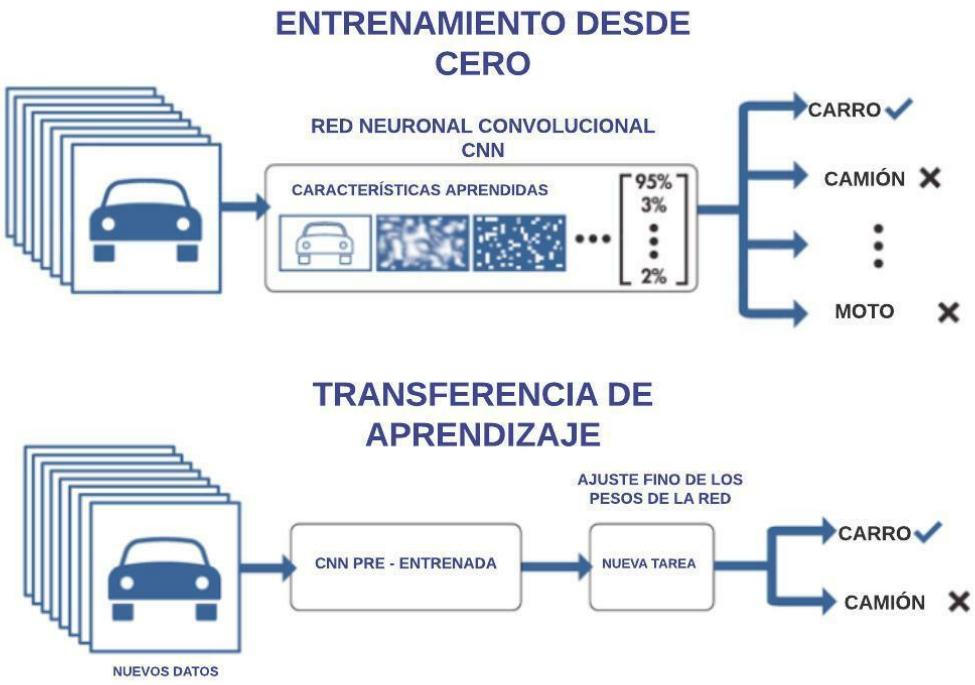
Software y Librerías	Versión instalada
Anaconda	4.12.0
CUDA	11.6
CuDNN	8.4.0
Python	3.6.13
Protobuf	3.20.0

Pre-entrenamiento de la red mediante la transferencia de aprendizaje

Uno de los inconvenientes presentados con el desarrollo de sistemas basados en Deep learning, radica en la cantidad de tiempo que implica entrenar una red neuronal artificial. Para optimizar este tiempo se utiliza la técnica de transferencia de aprendizaje, mediante la cual se transfiere el conocimiento aprendido para reconocer objetos inicializando la arquitectura con valores de un modelo pre-entrenado. Esta técnica permite utilizar modelos de CNN pre – entrenados sobre grandes bases de datos, y adaptarlos a un caso particular. En efecto, el modelo de red pre – entrenado que ha sido preparado para alguna tarea en general, es modificado para entrenar con datos propios. En la figura 32 se visualiza el entrenamiento de una CNN desde cero y con transferencia de aprendizaje (Shah, 2019).

Figura 32

Entrenamiento desde cero vs Transferencia de aprendizaje. (Shah, 2019)



En lo que respecta a las estrategias de la transferencia de aprendizaje, en la literatura se encuentran principalmente dos enfoques:

- Utilización de modelos pre – entrenados como extractores de características: En esta estrategia se congelan los pesos de toda la red, a excepción de la capa final. Sólo capa final es reconfigurada con los nuevos pesos aleatorios para satisfacer las condiciones del problema a solucionar.
- Ajustes de modelos pre – entrenados: Es una estrategia más compleja que la anterior, ya que no solo se reconfigura la capa final sino también otras capas de forma selectiva.

Resumiendo lo dicho, en la figura 33 se observan las dos estrategias: a) la utilización de modelos pre – entrenados como extractores de características, trabajando a nivel de la última capa de red. Y b) el ajuste de modelos pre – entrenados desarrollado a nivel de toda la red

Figura 33

Estrategias de transferencia de aprendizaje. (Pérez Lorenzo, 2019).

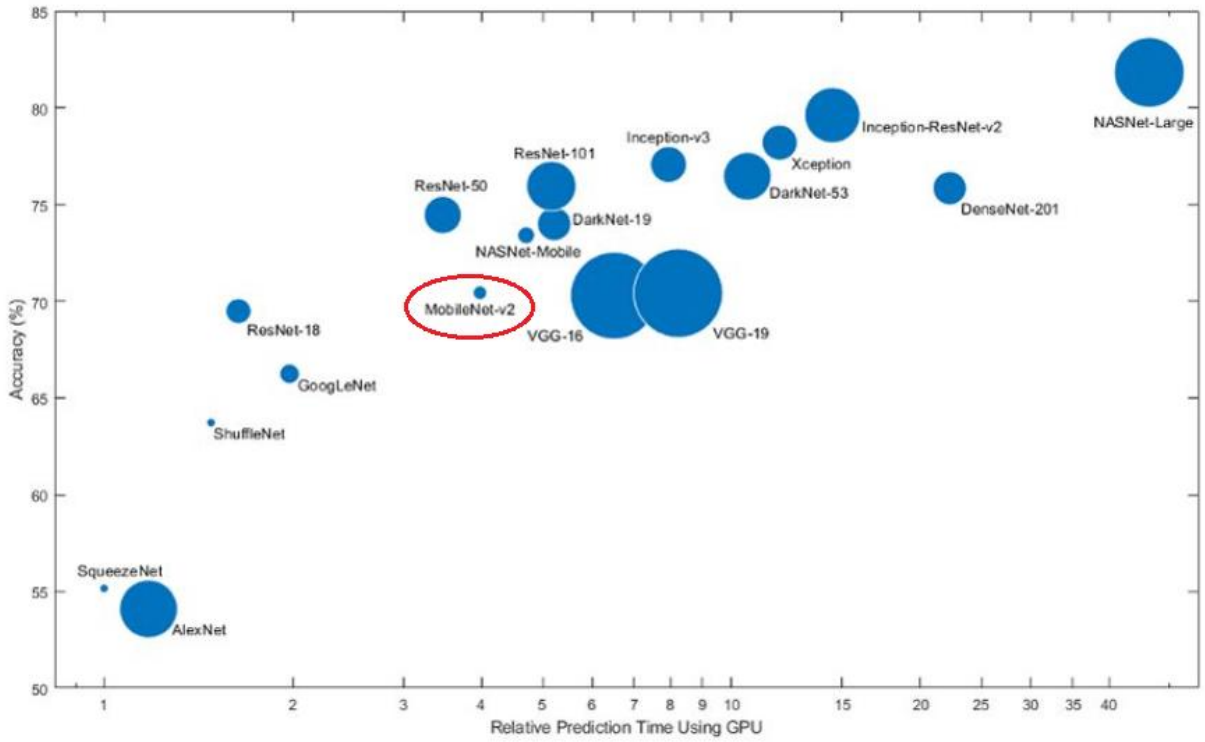
Modelos de CNN pre – entrenados para transferencia de aprendizaje

Una vez entendido los conceptos y estrategias de la transferencia de aprendizaje, es necesario tomar en cuenta algunas consideraciones para la selección del modelo de CNN pre – entrenado a utilizar. Entre ellos se debe considerar los siguientes:

- Tamaño: ¿Cuál es el espacio en memoria disponible para el modelo? Este factor reside directamente en el hardware donde se va a ejecutar el modelo, sea hardware integrado, PC portátil o de escritorio. Para este proyecto se considerará un modelo que pueda ser implementado en hardware embebido en un trabajo posterior.
- Precisión: ¿Qué tan bien funciona el modelo? Es un indicador del comportamiento del modelo frente al desarrollo de nuevas tareas de predicción
- Velocidad de predicción: ¿Cuál es la rapidez con la que el modelo hace predicciones a partir de nuevas entradas? Este parámetro puede variar en función del hardware utilizado y del tamaño del modelo. En otras palabras, un modelo de CNN de gran tamaño funcionara más lento en un hardware integrado que en un PC portátil o de escritorio.

Figura 34

Comparativa entre modelos de CNN pre – entrenados. (MathWorks, 2022).



Considerando estos factores, la figura 34 muestra la comparación de tamaño, precisión y velocidad de predicción entre varios modelos de CNN pre – entrenados. A partir de ellos, se ha seleccionado el modelo MobileNet – v2 de características mínimas en tamaño, con valores de precisión y velocidad de predicción aceptables (MathWorks, 2022).

Flujo de trabajo de la transferencia de aprendizaje

La transferencia de aprendizaje tiene gran variedad de aplicaciones y los modelos pre – entrenados poseen múltiples arquitecturas. Pero en forma general, la mayoría de técnicas de transferencia de aprendizaje siguen un proceso común, cuyos pasos se enlistan a continuación:

- Selección del modelo de CNN pre – entrenado.

- Reemplazar la última capa.
- Congelar los pesos (opcional).
- Entrenar el modelo para caso particular.
- Realizar predicciones y evaluar la predicción de la red.

Selección del modelo de CNN pre – entrenado: Como se mencionó anteriormente el modelo pre – entrenado elegido es MobileNet - v2. En razón de su reducido tamaño y que posee una precisión y velocidad de predicción aceptables. Este modelo tiene una arquitectura de red neuronal convolucional que se utiliza en hardware de bajo coste computacional, especialmente en dispositivos móviles, de allí su nombre. Dentro de la arquitectura de Mobilenet – v2 se encuentran 90 clases de objetos que fueron entrenados previamente en el set de datos COCO (Lin, y otros, 2021) y que son adaptados para las clases del nuevo conjunto de datos. Es decir que, la capa final compuesta por las 90 clases de objetos es reemplazada por otra capa de salida con las clases a detectar en el nuevo modelo de CNN.

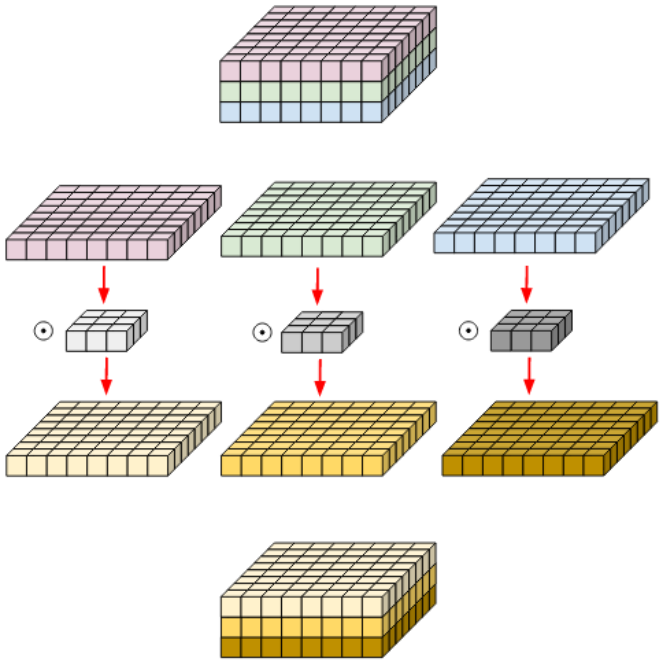
Mobilenet v2 es combinado con una arquitectura de detección de disparo único o en inglés Single Shot Detection (SSD). El SSD se basa en detectar múltiples objetos de una imagen en un solo disparo para lo cual incorpora en su arquitectura la predicción de clase y un cuadro delimitador en un solo proceso (Khandelwal, 2019). Estas características mejoran significativamente la velocidad de detección de objetos por parte de la CNN. Este modelo combinado denominado SSD - MobileNet v2 se caracteriza por tener dos módulos convolucionales.

- *El MobileNet v2* como red base que funciona a modo de extractor de características. Y que contiene un bloque adicional conocido como la convolución en profundidad o Depth wise convolution, que ayuda a evitar el sobreajuste de parámetros en el módulo principal. Fundamentalmente,

lo que hace la convolución en profundidad es dividir el filtro e imagen en canales diferentes y luego vuelve apilarlos, como se puede observar en la figura 35 (Pandey, 2018).

Figura 35

Convolución en profundidad o depth wise convolution



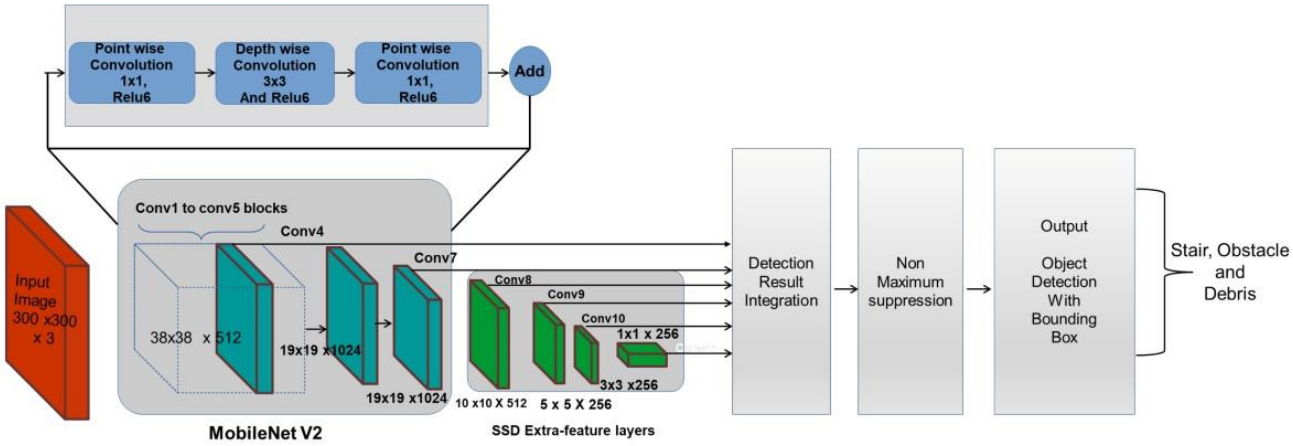
- *El SSD* como red integrada que funciona como localizador de objetos. Su arquitectura se basa en el uso de redes convolucionales que generan múltiples cuadros delimitadores correspondientes a las categorías de clase entrenadas. Cada instancia de clase de objeto detectada es estimada con un valor de probabilidad expresado en términos de porcentaje.

En la figura 36 se muestra la representación esquemática general del SSD MobileNet v2. En ella se puede observar el flujo de imágenes capturadas como entrada

del bloque de MobileNet v2. Este bloque genera un mapa de características que serán a su vez la entrada del bloque SSD. El SSD es el encargado de detectar la clase de un objeto y su ubicación mediante un cuadro delimitador (Ramalingam, y otros, 2021).

Figura 36

Esquema general del modelo SSD MobileNet v2. (Ramalingam, y otros, 2021).



En la tabla 7 se presenta las capas que componen a cada uno de los bloques del modelo SSD MobileNet V2, utilizado en este trabajo.

Tabla 7

Arquitectura utilizada del modelo SSD MobileNet V2. (Tsang, 2019), (QoChuk, 2022).

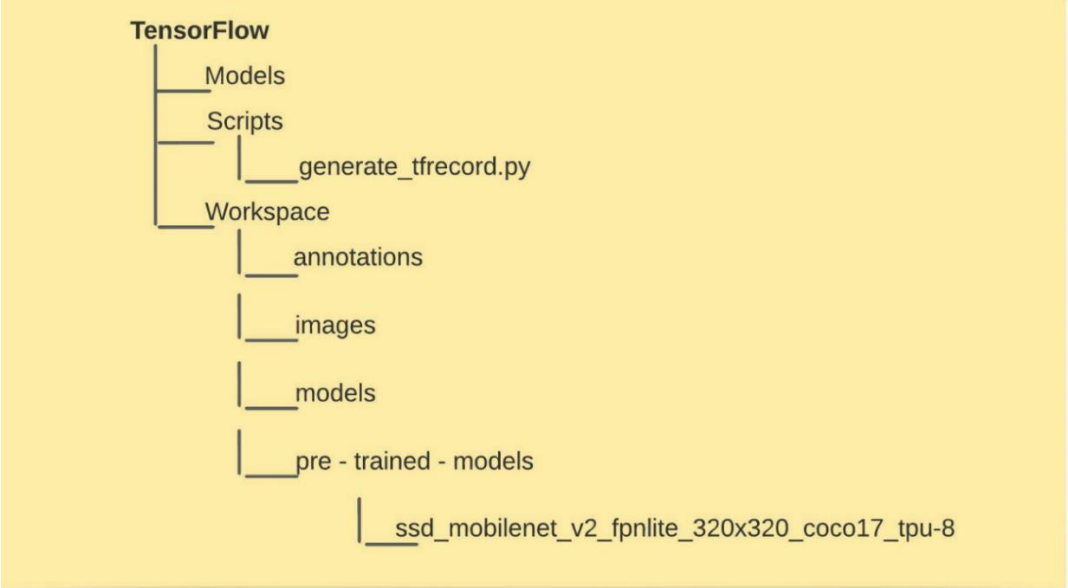
MobileNet V2	
Nombre y tipo de la capa	**Tamaño (Altura x ancho x profundidad)**
Convolución	320 x 320 x 3
Convolución en profundidad	160 x 160 x 32
Convolución en profundidad	160 x 160 x 16
Convolución en profundidad	80 x 80 x 24
Convolución en profundidad	40 x 40 x 32

Convolución en profundidad	20 x 20 x 64
Convolución en profundidad	20 x 20 x 96
Convolución en profundidad	10 x 10 x 160
Convolución	10 x 10 x 320
Avgpooling 10x10	10 x 10 x 1280
Convolución 1x1	1 x 1 x 1280
Single Shot Detection (SSD)	
Nombre y tipo de la capa	**Tamaño (Altura x ancho x profundidad)**
Convolución	3 x 3 x 1024
Convolución	1 x 1 x 1024
Convolución	1 x 1 x 256
Convolución	1 x 1 x 128
Convolución	1 x 1 x 128
Convolución	1 x 1 x 128

Una vez conocido en detalle el modelo de CNN pre – entrenado a utilizar. Se procede con el procedimiento de descarga y configuración de la CNN en el computador local. Para la descarga se visita el sitio web (Birodkar, Joglekar, & Rathod, 2021), donde se muestran varios modelos pre – entrenados en el conjunto de datos COCO 2017 (Lin, y otros, 2017). De preferencia se debe crear la carpeta con nombre “workspace” dentro del directorio donde se instaló la API de detección de objetos de tensorflow. Dentro de esta carpeta se ubicarán todos los archivos que se vayan generando en el desarrollo del trabajo. En la figura 37 se observa el directorio raíz, su estructura y los archivos que contiene.

Figura 37

Directorio de trabajo con archivo de modelo pre - entrenado.



En la figura 37 se observa que el modelo pre – entrenado descargado se ubica dentro de la carpeta “pre – trained – models”. El resto de carpetas se explicarán conforme se vayan utilizando en el documento.

Reemplazar la última capa: En este paso se reconfigura la capa final de la red, según el número de clases en particular. Cabe señalar que el modelo SSD de MobileNet v2 se entrenó originalmente con 90 categorías de clase, para este proyecto ese modelo fue cambiado en su capa final. Teniendo presente que las clases con las que se reentrena el modelo son cinco: persona parada, agachada, acostada, sentada y caída.

Congelar los pesos (opcional): Este paso es opcional y depende en gran medida del tamaño del conjunto de datos con que se entrene el modelo. Al congelar los pesos de las capas anteriores de la red, los parámetros de las mismas no se actualizan. De modo que, el entrenamiento de la red puede acelerar significativamente.

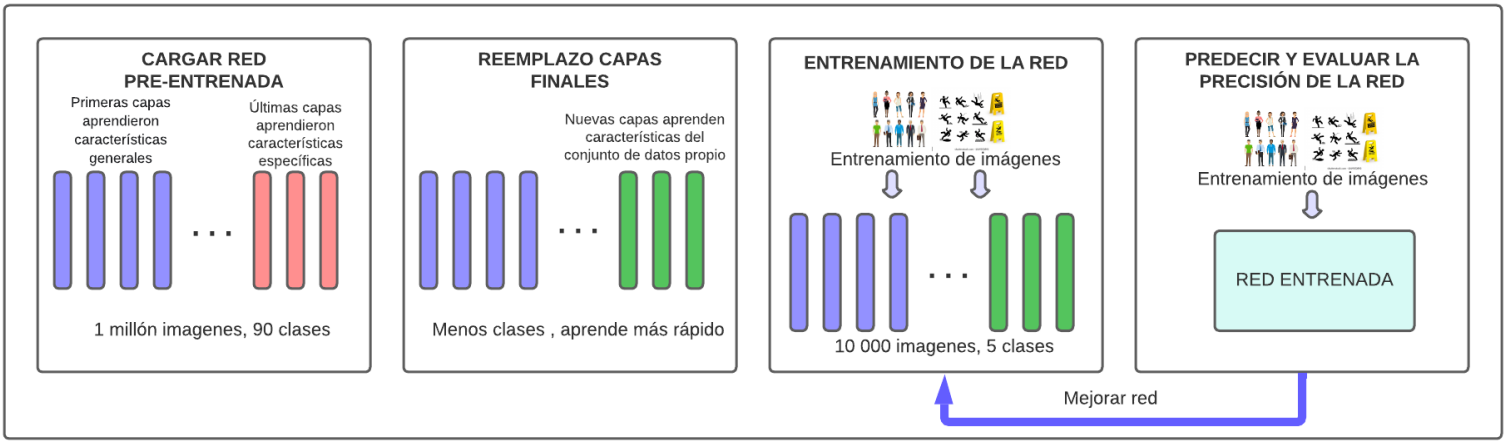
Entrenar el modelo para caso particular: El nuevo entrenamiento capacitará al modelo de red para que aprenda e identifique las características asociadas al conjunto de datos y categorías de clase, del caso en particular. En efecto, el entrenamiento para transferencia de aprendizaje necesita de menos datos y tiempo que, para un entrenamiento de modelo de red desde cero. Este procedimiento se detalla en el apartado de preparación del set de datos de entrenamiento.

Realizar predicciones y evaluar la predicción de la red: Una vez que la red ha sido entrenada con los nuevos datos, está en capacidad de realizar nuevas predicciones en base a las categorías de clase en particular. La evaluación de predicción del modelo se determina entorno a su funcionamiento y capacidad de realizar nuevas predicciones.

En la figura 38 se encuentra plasmada gráficamente la metodología de la aplicación de la técnica de transferencia de aprendizaje. En la figura se puede visualizar desde el paso inicial, selección del modelo pre – entrenado, hasta el último paso, que constituye la evaluación de la precisión del modelo, en base a su funcionamiento y comportamiento frente a nuevas predicciones.

Figura 38

Flujo de trabajo de la transferencia de aprendizaje.

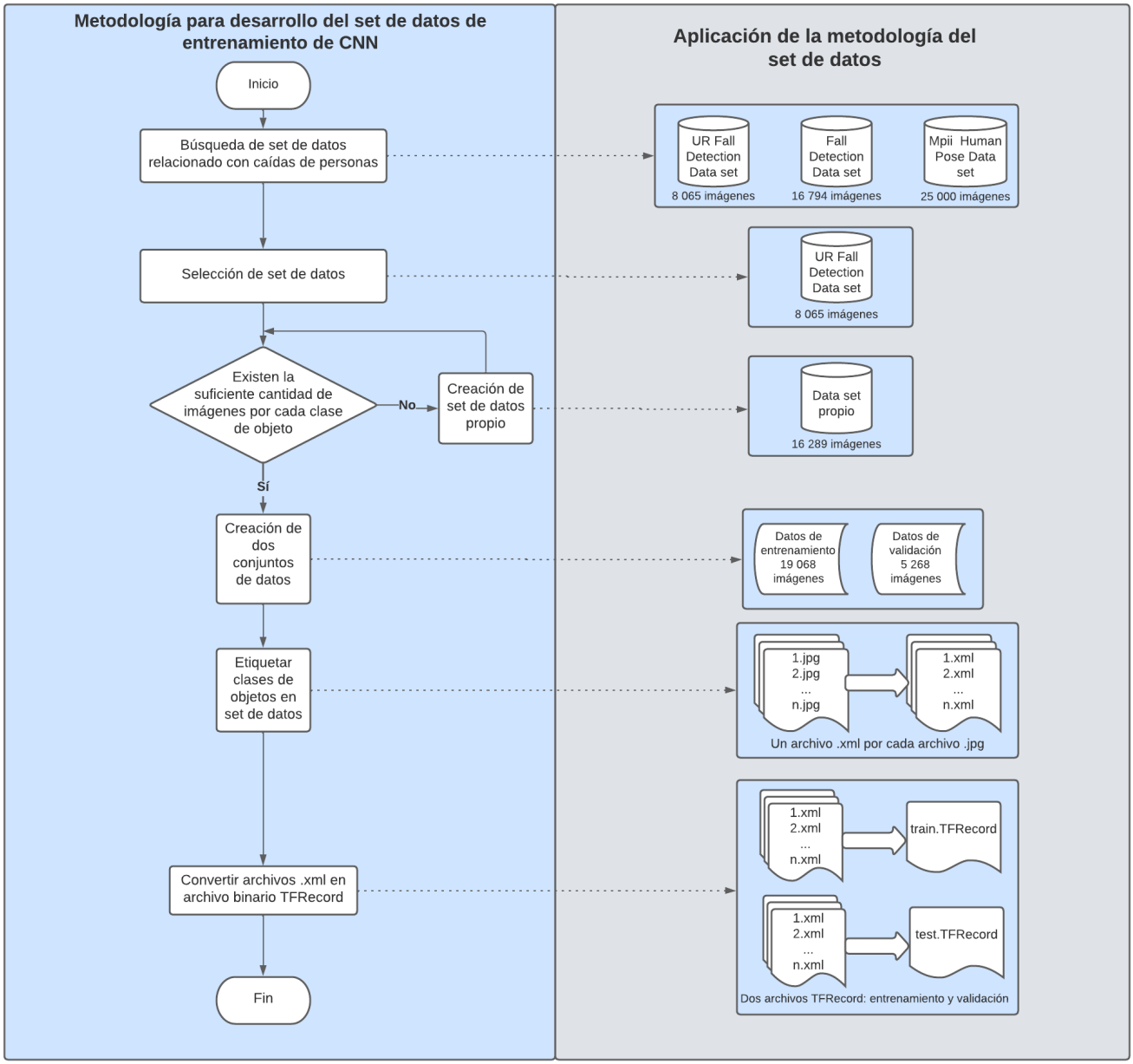


Metodología de desarrollo del set datos de entrenamiento

La metodología de desarrollo del set de datos de entrenamiento, es un conjunto de mecanismos o procedimientos que permiten definir la información que debe procesar y aprender el modelo de CNN. La figura 39 ilustra los pasos de la metodología propuesta.

Figura 39

Metodología y aplicación del set de datos de entrenamiento.



Preparación del set de datos de entrenamiento

El set de datos de entrenamiento es un conjunto de imágenes seleccionadas previamente, con el objetivo de recabar características especiales a ser aprendidas por el modelo pre - entrenado. Para el presente proyecto, las imágenes deben mostrar a personas realizando sus diferentes actividades diarias. Dado que este proyecto busca determinar una condición o posición de persona caída, el clasificador de la red debe clasificar las distintas posiciones de las personas, es decir, personas paradas, sentadas, agachadas, acostadas o caídas. Esta son las clases de objetos a ser tomadas en cuenta por la CNN. A continuación, se describe la metodología utilizada para el desarrollo del set de datos de entrenamiento del modelo SSD MobileNet V2.

Descripción del set de datos utilizado

El conjunto de datos de este trabajo es creado en base a un set de datos alojado en la web (UR Fall Detection Dataset) y un set de datos propio. Señalando que los dos sets de datos contienen imágenes de varias personas, en las posturas correspondientes a las clases de objetos antes indicadas. En el caso de los datos provenientes de la UR Fall Detection Dataset estos son descargados de la página web donde están disponibles (Kępski & Kwolek, 2014). En tanto que el set de datos propios ha sido captado mediante una aplicación desarrollada en lenguaje Python que toma fotografías utilizando la cámara del computador local. En el anexo 1 se muestra el código de la aplicación utilizado para este propósito. Es importante señalar que el conjunto de datos debe ser uniforme, es decir todas las imágenes deben tener el mismo ancho x altura x profundidad. Para el caso en particular las imágenes de los dos sets de datos tienen las dimensiones 640 x 480 x 3. Debido a que la cantidad de imágenes del set de datos UR Fall Detection, no era

suficiente, se duplicaron las imágenes captadas para el proyecto. La tabla 8 resume el número de imágenes utilizadas por cada set de datos.

Tabla 8

Cantidad de imágenes por set de datos.

Set de datos	**Número de imágenes**
UR Fall Detection Dataset	8 065
Set de datos propio	16 289
Total	**24 354**

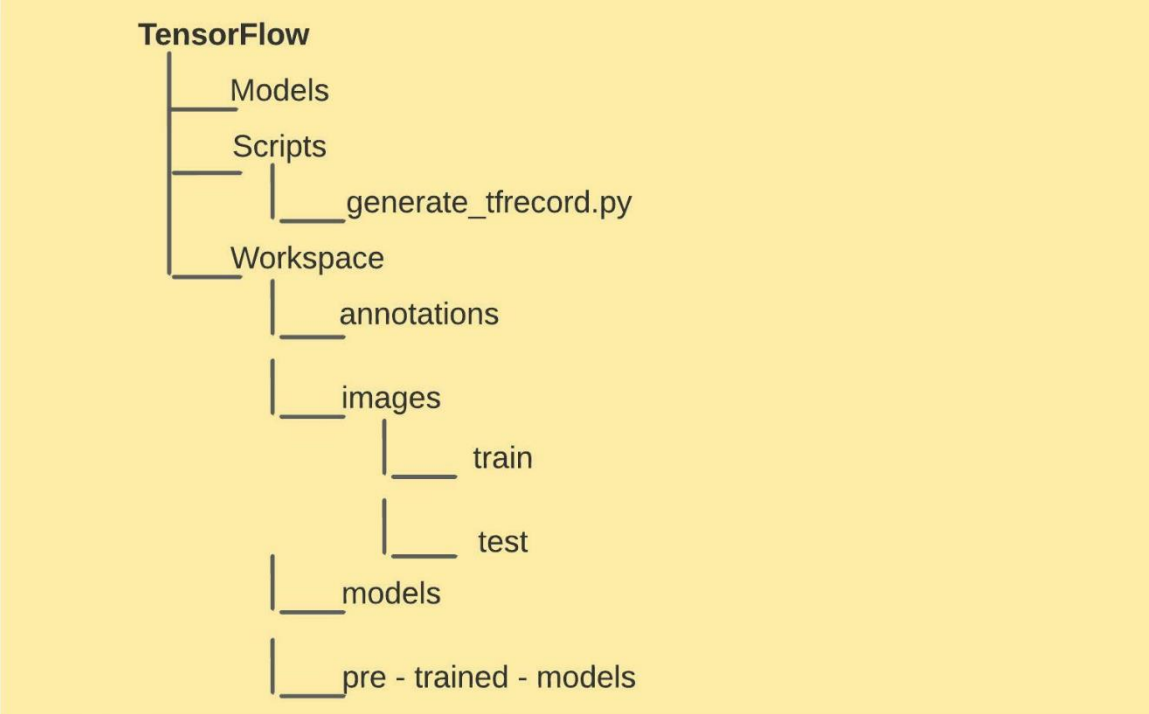
Cabe destacar que, una vez completado el set de datos se particiona en dos subconjuntos, el de entrenamiento y el de validación. El de entrenamiento es el que se usa para entrenar el modelo de CNN, y de donde se determinan las características específicas que describen cada instancia de clase. En tanto que el set de validación es utilizado para evaluar el entrenamiento del modelo de CNN.

Para la división de los subconjuntos es aconsejable tomar 80 – 90 % del set de datos para entrenamiento y el 10 – 20 % restante para validación (De Luca & Irigoitia, 2021). Tomando en cuenta estas sugerencias dadas en la literatura, el subconjunto de entrenamiento para este proyecto quedó formado por 19 068 imágenes mientras que el de validación por 5 268 imágenes.

En el directorio de trabajo dentro de la ubicación C:\Users\TensorFlow\workspace\images se asignan dos carpetas “train” y “test” que permiten guardar las imágenes de entrenamiento y validación respectivamente. En la figura 40 se observa el directorio raíz con las carpetas antes mencionadas.

Figura 40

Directorio de trabajo con archivo train y test del set de datos.

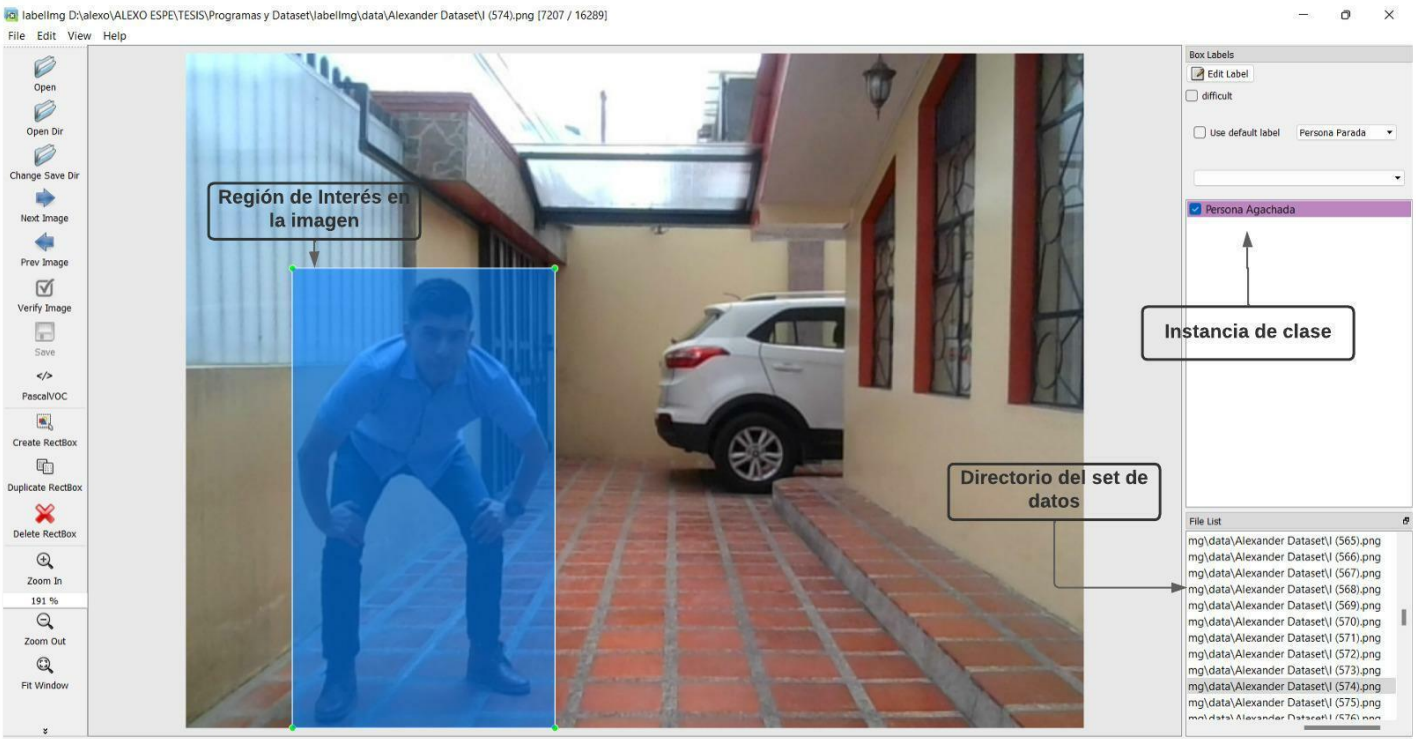


Edición del set de datos

La edición del set de datos consiste en el etiquetado de las regiones de interés en cada imagen. Identificando las instancias de clases a ser reconocidas por el modelo de CNN. Para este fin se trabaja con LabelImg, un programa de anotación gráfica de imágenes, donde se pueden marcar regiones de interés correspondientes a las instancias de clase. LabelImg proporciona una interfaz gráfica y es escrita en lenguaje Python. Cabe indicar que la información de anotación en cada imagen es convertida en archivo XML. En la figura 41 se muestra la ventana principal de trabajo de la herramienta LabelImg. Mostrando la región de interés a ser entrenada por el modelo de CNN y la instancia de clase que le corresponde.

Figura 41

Etiquetado de imágenes en LabelImg.



Creación de archivos para entrenamiento de CNN

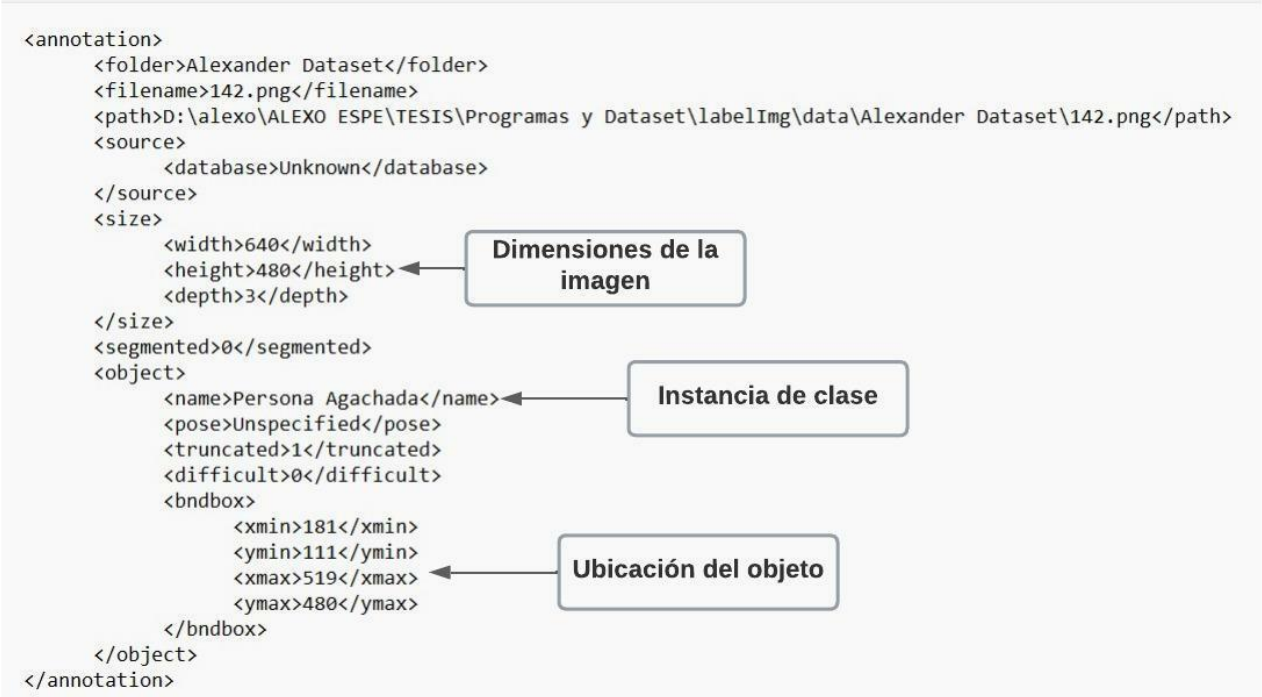
Posterior al proceso de etiquetado de imágenes del set de datos, empieza la preparación de archivos que dan lugar al entrenamiento del modelo de CNN. Entre los formatos de archivo con los que se trabaja, se encuentran: XML, PBTXT y TFRecords.

Archivos XML

Es un tipo de archivo que contiene información de la imagen etiquetada. Dicha información contiene la ubicación y la clase a la que pertenece cada objeto del set de datos. En la carpeta “train” y “test” se encuentra un archivo XML por cada imagen. La estructura del archivo XML se ilustra en la figura 42.

Figura 42

Estructura del archivo XML.




```
<annotation>
      <folder>Alexander Dataset</folder>
      <filename>142.png</filename>
      <path>D:\alexo\ALEXO ESPE\TESIS\Programas y Dataset\labelImg\data\Alexander Dataset\142.png</path>
      <source>
            <database>Unknown</database>
      </source>
      <size>
            <width>640</width>
            <height>480</height>
            <depth>3</depth>
      </size>
      <segmented>0</segmented>
      <object>
            <name>Persona Agachada</name>
            <pose>Unspecified</pose>
            <truncated>1</truncated>
            <difficult>0</difficult>
            <bndbox>
                  <xmin>181</xmin>
                  <ymin>111</ymin>
                  <xmax>519</xmax>
                  <ymax>480</ymax>
            </bndbox>
      </object>
</annotation>
```

Archivo PBTXT

El archivo PBTXT o también conocido como Label Map contiene la configuración con las instancias de clase que el modelo de CNN debe entrenar. Por lo que las etiquetas de imagen creadas en la herramienta LabelImg, deben ser las mismas que en el archivo PBTXT. El código en Python utilizado para crear el archivo de configuración PBTXT, se encuentra en el anexo 2.

En la figura 43 se puede observar las cinco instancias de clase utilizadas para este trabajo. Teniendo en cuenta que, cada nombre en el archivo PBTXT debe corresponderse de la misma forma con las etiquetas de imagen en LabelImg. En otras palabras, los nombres de clase deben estar escritos de la misma forma sea en mayúsculas, minúsculas o espacios.

Figura 43

Archivo LabelMap con las instancias de clase definidas.

```
item {
      name:'Persona Acostada'
      id:1
}
item {
      name:'Persona Agachada'
      id:2
}
item {
      name:'Persona Caida'
      id:3
}
item {
      name:'Persona Parada'
      id:4
}
item {
      name:'Persona Sentada'
      id:5
}
```

Archivos TFRecords

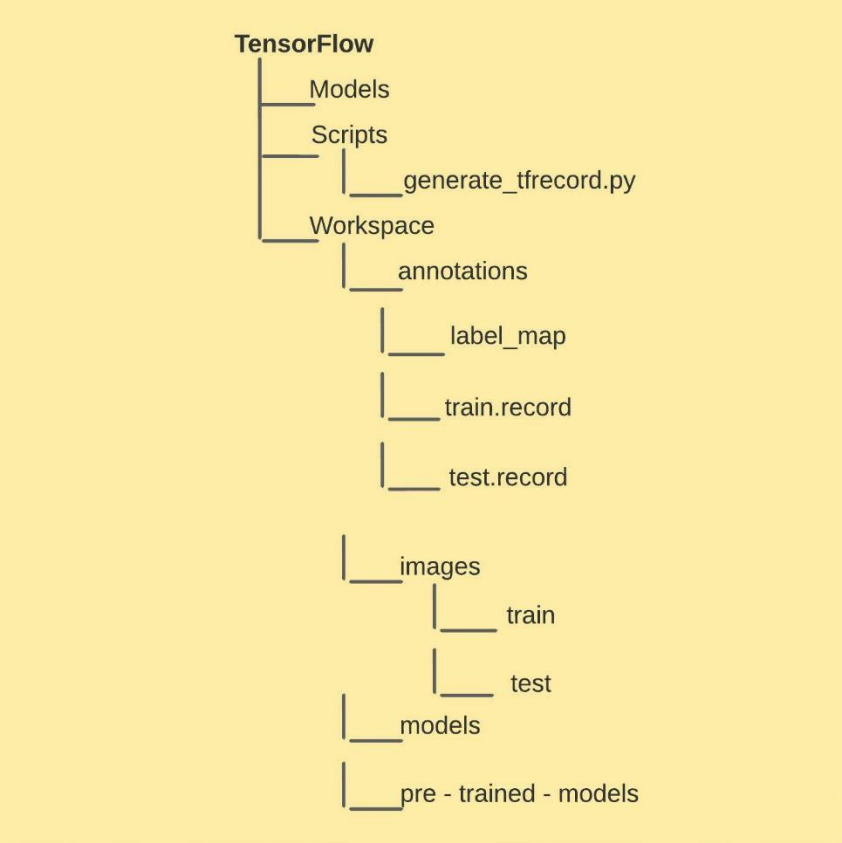
Como se indicó anteriormente, la cantidad de información proveniente del set de datos para realizar el entrenamiento de un modelo de CNN es notablemente grande. En consecuencia, se presentan problemas en el espacio de almacenamiento y tiempo de entrenamiento del modelo. Por lo que los archivos TFRecords definidos como archivos binarios están orientados a optimizar la cantidad de información para llevar los registros.

De allí que, para la optimización de recursos en el proceso de entrenamiento de la red, se convierte los archivos XML en archivos TFRecords. Siendo importante mencionar que se crean únicamente dos archivos TFRecords, uno correspondiente a la carpeta Train de entrenamiento y otro a la carpeta test de validación. En el Anexo 2 y 3 se encuentra el código para convertir los archivos XML en formato TFRecords. Como se

muestra en la figura 44, dentro de la carpeta “annotations” del directorio raíz se encuentran los archivos label_map y TFRecords, con los cuales se realiza el entrenamiento del modelo de CNN.

Figura 44

Directorio de trabajo con archivo train y test record.



En el anexo 9 el lector puede encontrar más información sobre el almacenamiento en disco y tiempo requerido para realizar el set de datos de este trabajo.

Entrenamiento y parámetros

El entrenamiento es la fase de aprendizaje de las redes neuronales artificiales. Aquí es donde el modelo de CNN se capacita para reconocer características generales y específicas de cada objeto a detectar. ¿Pero cómo funciona esto? Básicamente, el proceso de entrenamiento radica en la actualización de pesos de todas las neuronas de

la red, mediante un algoritmo conocido como propagación hacia atrás o backpropagation. Antes de explicar el algoritmo en mención es necesario definir los siguientes parámetros de entrenamiento.

Función de pérdida (Loss Function)

La función de pérdida también conocida como función de costo es una métrica que permite evaluar la desviación entre las predicciones realizadas por el modelo de CNN y los valores reales de los datos de entrenamiento. La función de pérdida debe ser una función logarítmica decreciente que tiende a cero. Concluyendo que, un valor bajo en esta función es indicador de que la CNN tiene un buen desempeño. Y por el contrario un valor alto en esta función determina que la CNN es ineficiente (Rodríguez, 2018).

Tasa de aprendizaje (Learning rate)

La tasa de aprendizaje es un parámetro que mide el porcentaje de cambio con el que se actualizan los pesos de las neuronas de la CNN. En otras palabras, es la rapidez con que la red neuronal renueva los conocimientos aprendidos. El rango de valores de este parámetro se encuentra entre 0 y 1. Cabe mencionar que tasas de aprendizaje más pequeñas toma más tiempo de entrenamiento. En tanto que en tasas de aprendizaje con valores cercanos a 1, el entrenamiento es más rápido pero los pesos finales no son óptimos. Por lo que los valores adecuados para un entrenamiento eficaz de la CNN, son cercanos a 0.1 y 0.01 (Flórez López & Férnandez Férnandez, 2008).

Época (Epoch)

Época se define como una revisión o pasada que realiza el modelo de CNN, por todo el conjunto de datos para su entrenamiento. Conocido también como iteración y relacionado con el parámetro tamaño del lote (batch size).

Tamaño del lote (Batch size)

Es el número de datos que la CNN entrena en cada iteración o época. Y permite optimizar los recursos computacionales en memoria, ya que no se necesita cargar todo el conjunto de datos sino únicamente el tamaño del lote asignado, por cada iteración.

Precisión (Accuracy)

Es el porcentaje de predicciones correctas que realizo el modelo de CNN según la información proporcionada por el set de datos. Matemáticamente la precisión tiene la siguiente definición:

$$Precisión\ (\%) = \frac{Número\ de\ predicciones\ correctas}{Número\ total\ de\ predicciones} x100$$

El algoritmo de propagación hacia atrás es el encargado de optimizar la función de pérdida para mejorar la tarea de predicción de objetos de la red neuronal. Este algoritmo calcula el error y lo difunde hacia atrás a cada una de las capas para minimizar el valor de la función de pérdida.

Finalmente, para realizar el proceso de entrenamiento del modelo de CNN se ejecuta el código mostrado en la figura 45 dentro de la carpeta raíz de trabajo C:\Users\TensorFlow. El código de Python completo de la transferencia de aprendizaje con el modelo de CNN empleado en este trabajo y la importación de archivos de entrenamiento se encuentran en el anexo 4.

Figura 45

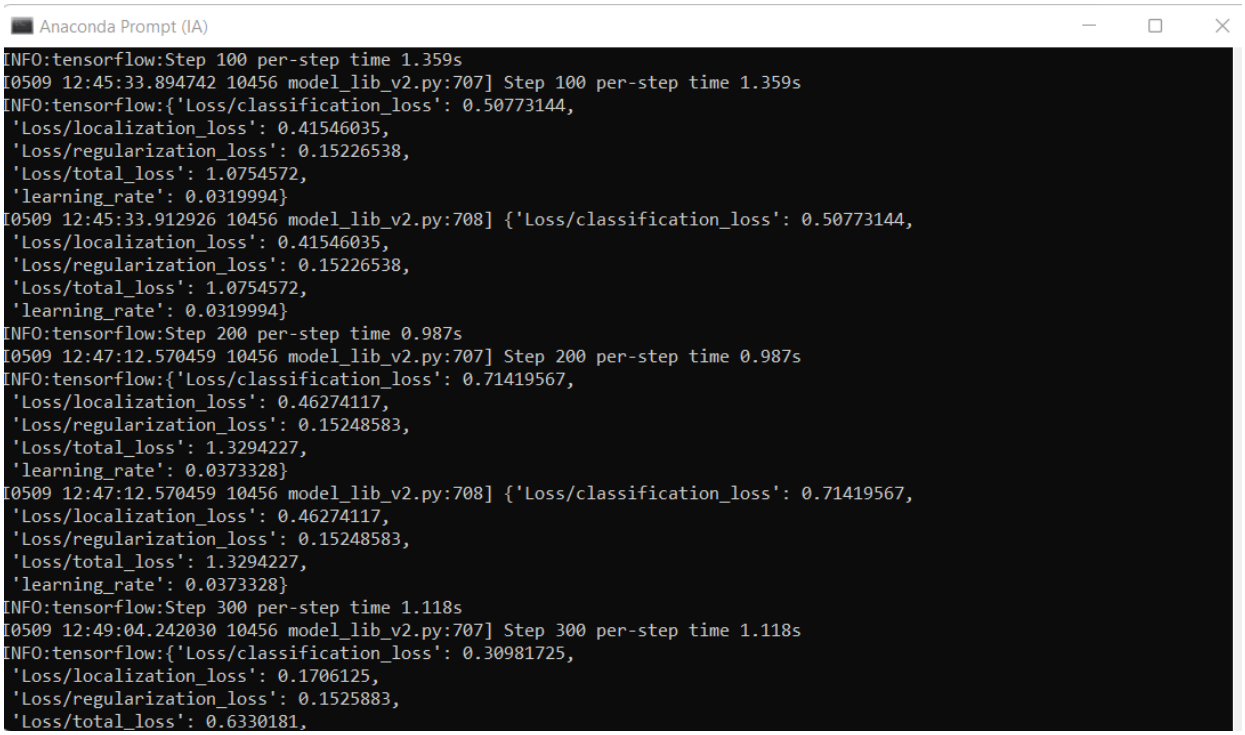
Código de entrenamiento para CNN.

```
print("""python {}/research/object_detection/model_main_tf2.py --
model_dir={}/{} --pipeline_config_path={}/{}/pipeline.config --
num_train_steps=45000""".format(APIMODEL_PATH,
MODEL_PATH,CUSTOM_MODEL_NAME,MODEL_PATH,CUSTOM_MODEL_NAME))
```

La figura 46 muestra el inicio del entrenamiento de la CNN, en sus dos pasos iniciales 100 y 200. El parámetro que se debe minimizar y con tendencia a cero es la función de pérdida, o denotada en el entrenamiento como “Loss/classification_loss”. Inicialmente esta función de pérdida se encuentra con valores de 0.5077 y 0.7141, los cuales indican que la CNN no se encuentra aún capacitada para realizar predicciones.

Figura 46

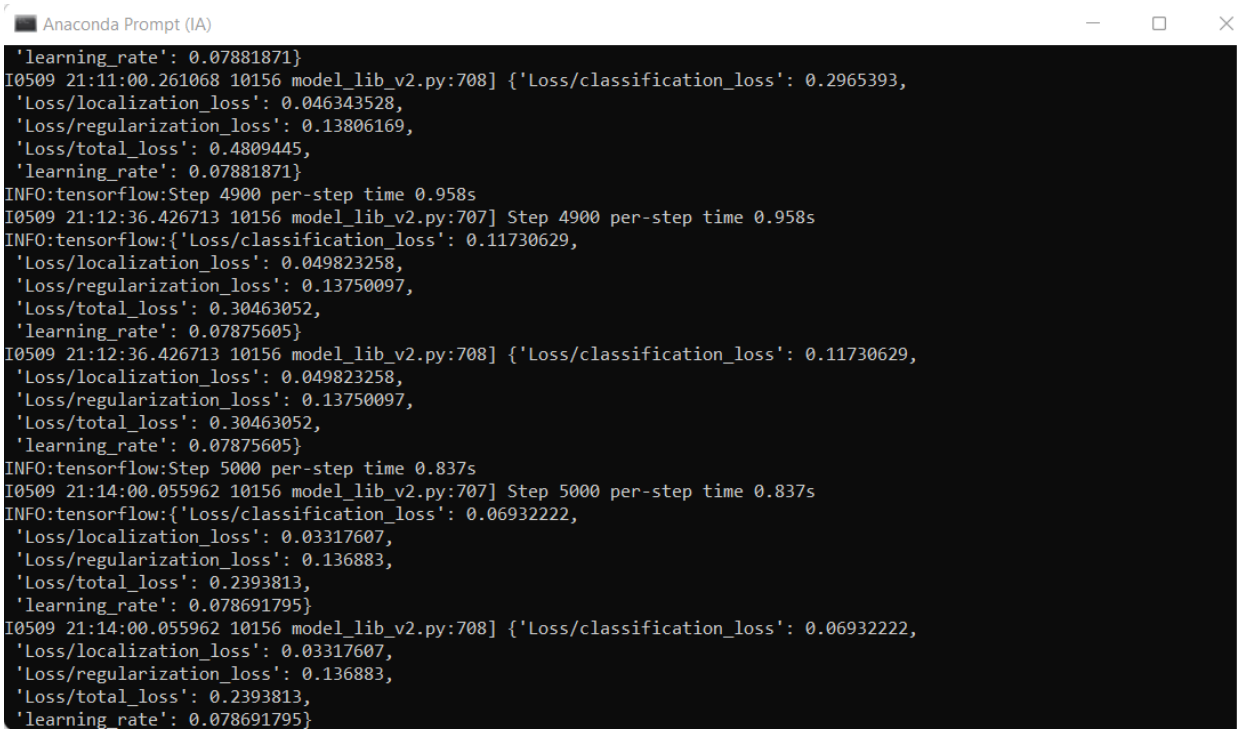
Entrenamiento de la CNN - Paso 100 - 200



En la figura 47 se puede observar como conforme se entrena más la CNN, la función de pérdida tiende a cero. Con pasos de entrenamiento de 4900 y 5000 se puede observar que la función de pérdida tiene un valor de 0.1173 y 0.0693 respectivamente. Destacando que la CNN se encuentra capacitada para iniciar predicciones para las que fue entrenada. Ya que la función de pérdida es menor que el 10% o 0.10.

Figura 47

Entrenamiento de la CNN - Paso 4900 - 5000



Envío de notificación de alerta de caída a plataforma de mensajería

Una vez realizado el proceso de entrenamiento se va a probar su efectividad mediante la verificación de las predicciones realizadas por el modelo de CNN, y su respectiva probabilidad de vinculación a cada instancia de clase. En el anexo 5 se muestra el código en Python utilizado para hacer las predicciones del modelo de CNN entrenado. Los resultados de estas predicciones se muestran en el siguiente capítulo.

Finalmente, el objetivo del sistema es alertar sobre una posible caída a otras personas que puedan brindar auxilio a la persona que sufrió el percance. Para este propósito de notificación, se ha previsto que el sistema envíe un mensaje por medio de la plataforma de mensajería instantánea Whatsapp a un contacto configurado previamente. La importancia de este mensaje radica en que se pueden tomar medidas de auxilio

rápidamente ante una alerta de la ocurrencia de este evento. En la figura 48 y 49 se muestra la notificación por mensaje de WhatsApp y código en Python de la alerta de caída respectivamente.

Figura 48

Envío de mensaje por WhatsApp ante detección de caída.

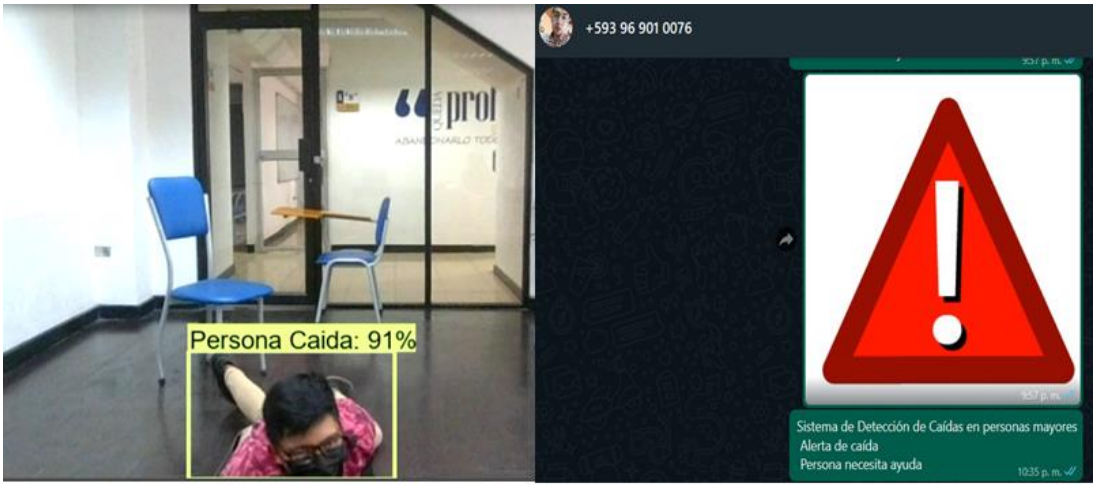


Figura 49

Código de envío de mensaje por WhatsApp ante alerta de caída.

```
import pywhatkit

class_objetos = []
contador = 0
aux = 1


class_objetos = detections['detection_classes']+label_id_offset

contador = contador + 1

#Enviar mensaje de Whatsapp
if class_objetos [0] == 3 and aux == 1 and contador > 50:
    aux = 0
    contador = 0
    pywhatkit.sendwhatmsg_instantly("+593969010076", "Sistema de Detecc
de Caídas en personas mayores\n Alerta de caída\n Persona necesita ayud
    pywhatkit.sendwhats_image("+593969010076", "Alerta.png")


if class_objetos [0] != 3 and aux == 0:
    aux = 1
```

Capítulo IV. Experimentación y resultados

Hasta aquí se ha explicado la configuración y puesta a punto de la plataforma del sistema de inteligencia artificial utilizando la técnica del Deep learning que se utilizó para la realización del presente proyecto. En este apartado se dedica a las pruebas de utilización del sistema para la detección de caídas de personas. Adicionalmente se realizará el estudio y análisis de las métricas de entrenamiento, que permiten evaluar la eficiencia del modelo de CNN. Para cumplir con este objetivo se utilizará herramientas alojadas en la web que presentan datos estadísticos relacionados con la función de pérdida, tasa de aprendizaje y pesos de las neuronas.

Tensorboard

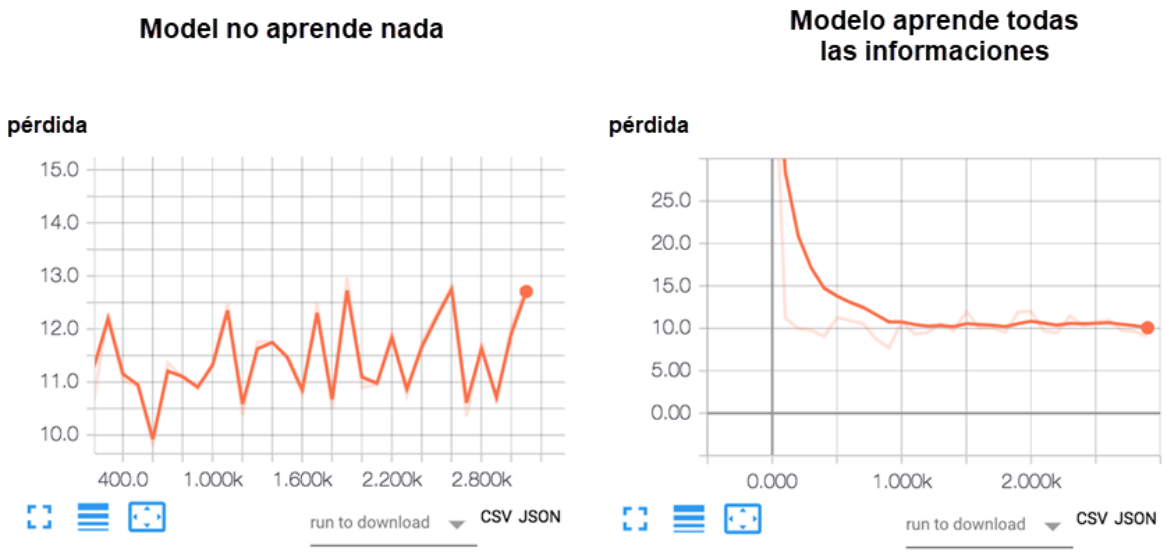
La herramienta Tensorboard es una extensión de Tensorflow que evalúa y depura modelos de CNN. Para el modelo entrenado en este trabajo, SSD Mobilenet V2, se consideran los pasos realizados de entrenamiento. Siendo en total 45000 pasos tomados de 100 en 100.

Función de pérdida evaluada

Al analizar la función de pérdida cabe destacar que si su gráfica tiene una tendencia cercana a cero el modelo aprende. en tanto que, si la función de pérdida no tiene este comportamiento, es motivo de algún fallo en el entrenamiento del modelo. La figura 50 muestra los dos tipos de conducta en la función de pérdida (Guru99, 2022).

Figura 50

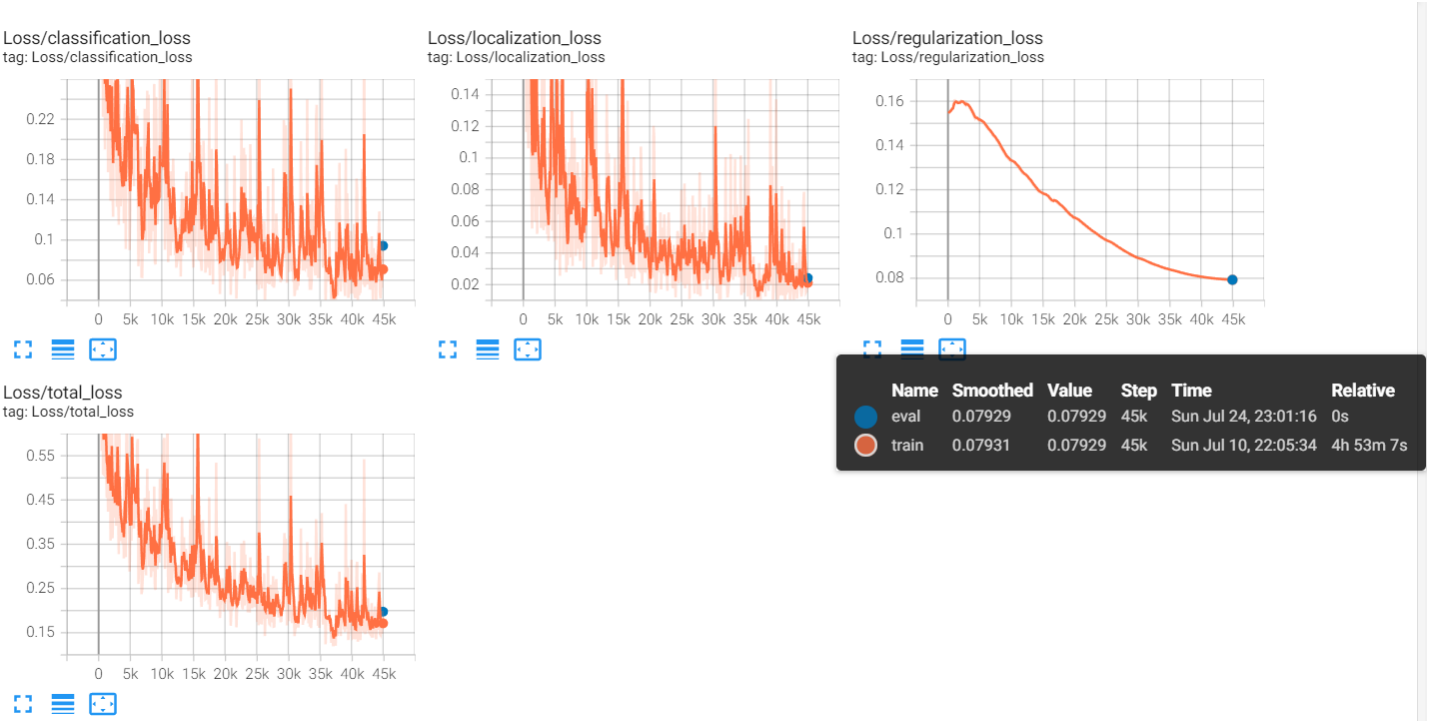
Comportamiento de la función de pérdida. (Guru99, 2022).



Al aplicar la herramienta que realiza las métricas de la función de pérdida sobre el sistema configurado se puede observar los resultados obtenidos en la figura 51. En ella, se puede apreciar que, a pesar de la presencia de algunas oscilaciones la función de pérdida tiene una tendencia a cero, a medida que avanza el entrenamiento. Esto demuestra un correcto entrenamiento de la CNN.

Figura 51

Función de pérdida del modelo SSD Mobilenet V2 entrenado.

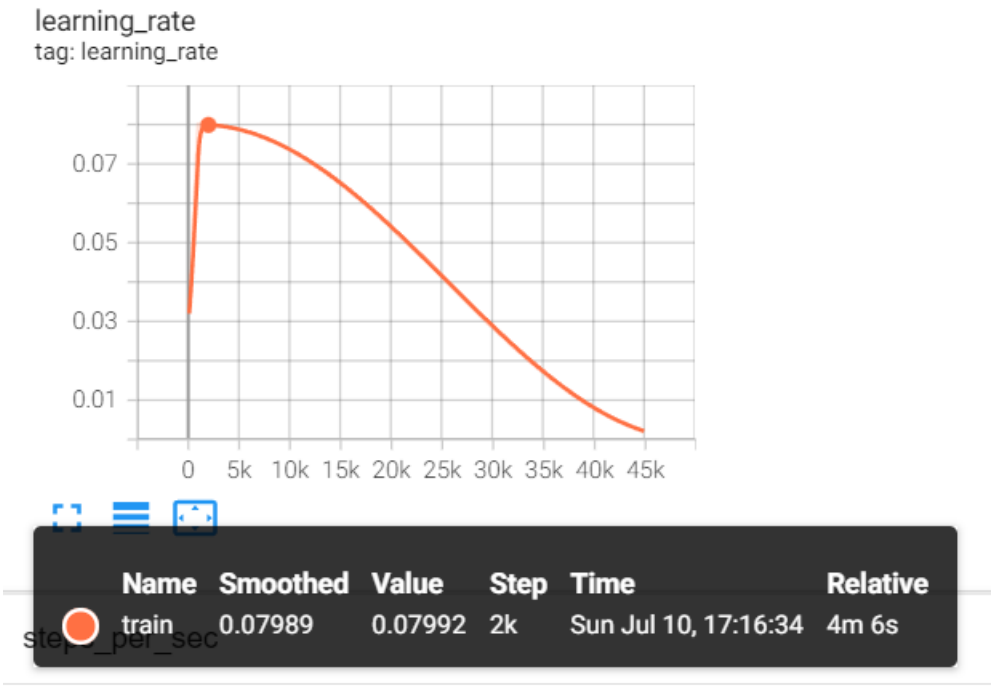


Tasa de aprendizaje evaluada

Otra métrica a evaluarse es la tasa de aprendizaje, que tiene su valor máximo en los primeros pasos del entrenamiento, específicamente a los 2000 pasos. Como se puede observar en la figura 52, en los pasos finales del entrenamiento esta tasa de aprendizaje está por debajo del 0.01. Cabe señalar que en pasos finales la tasa decrementa más lentamente.

Figura 52

Tasa de aprendizaje en el modelo SSD Mobilenet V2.



Ejecución del modelo de CNN con datos reales

Para la puesta en marcha del sistema de detección de caídas en personas mayores, se hicieron varias pruebas con varias personas en diferentes entornos. En las figuras 53 hasta la figura 70 se muestran algunos ejemplos de las predicciones realizadas por el modelo de CNN.

Figura 53

Predicción en tiempo real de las clases: Persona Sentada 62% y Parada 73%.

Figura 54

Predicción en tiempo real de las clases: Persona Sentada 56% y Agachada 60%.

Figura 55

Predicción en tiempo real de la clase: Persona Parada 70%.

Figura 56

Predicción en tiempo real de la clase: Persona Agachada 89%.

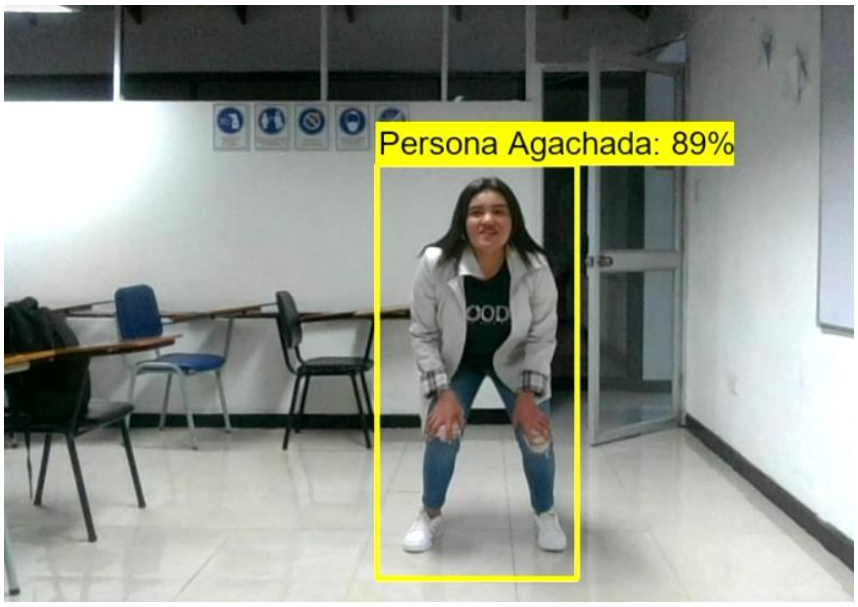


Figura 57

Predicción en tiempo real de la clase: Persona Caída 86%.

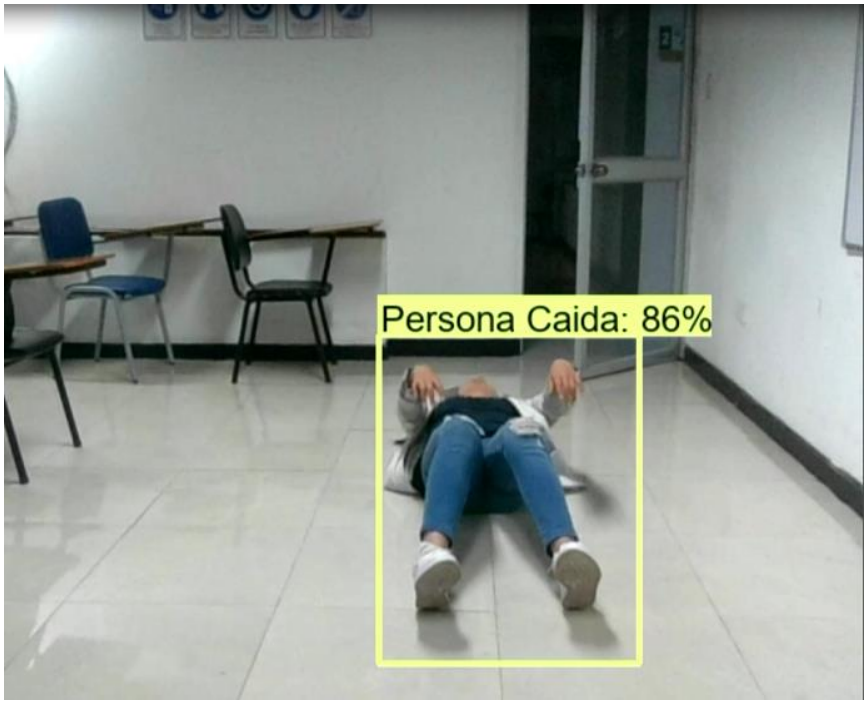


Figura 58

Predicción en tiempo real de la clase: Persona Agachada 68%.

Figura 59

Predicción en tiempo real de la clase: Persona Sentada 93%.

Figura 60

Predicción en tiempo real de la clase: Persona Caída 89%.

Figura 61

Predicción en tiempo real de la clase: Persona Parada 65%.

Figura 62

Predicción en tiempo real de la clase: Persona Agachada 73%.

Figura 63

Predicción en tiempo real de la clase: Persona Agachada 90%.

Figura 64

Predicción en tiempo real de la clase: Persona Sentada 79%.

Figura 65

Predicción en tiempo real de la clase: Persona Sentada 94%.

Figura 66

Predicción en tiempo real de la clase: Persona Caída 98%.

Figura 67

Predicción en tiempo real de la clase: Persona Caída 66%.

Figura 68

Predicción en tiempo real de la clase: Persona Caída 62%.

Figura 69

Predicción en tiempo real de las clases: Persona Parada 56% y Agachada 65%.

Figura 70

Predicción en tiempo real de las clases: Persona Caída 61% y Agachada 63%.

Evaluaciones

Las evaluaciones permiten determinar que tan bien está funcionando el sistema de detección de caídas en personas. Tales evaluaciones se realizaron en 15 personas diferentes, hombres y mujeres de varias edades que oscilan entre 17 – 65 años. Para evaluar la precisión del modelo de CNN se consideró diferentes ambientes (Hogar, aire libre y lugares de trabajo).

En la evaluación se encuentran cuatro resultados posibles:

- Verdadero positivo (VP): El valor real y la predicción son positivos.
- Verdadero negativo (VN): El valor real y la predicción son negativos.
- Falso positivo (FP): El valor real es negativo pero la predicción es positiva.
- Falso negativo (FN): El valor real es positivo pero la predicción es negativa.

En total se registraron dos evaluaciones de dos minutos cada una. En cada evaluación se reconocieron 10 predicciones por persona. Para lo cual se definen ambas métricas de la siguiente manera:

La exactitud representa las predicciones positivas marcadas como correctas. Matemáticamente la exactitud se calcula con la siguiente fórmula:

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

La precisión representa el porcentaje de predicciones positivas detectadas. Se representa como la fracción de resultados verdaderos positivos con respecto a todos los resultados verdaderos o falsos positivos. La precisión se representa con la siguiente fórmula:

$$Precisión = \frac{VP}{VP + FP}$$

Resultados obtenidos de la evaluación

En la Tabla 9 se muestran los resultados obtenidos en las evaluaciones de cada persona y los cálculos parciales de la exactitud y precisión en cada caso.

Tabla 9

Resultados de evaluación del sistema de detección de caídas en personas.

Persona	Edad	Sexo	VP	VN	FP	FN	TOTAL	Exactitud (%)	Precisión (%)
1	17	Masculino	8	8	4	0	20	80	66.67
2	18	Femenino	9	9	2	0	20	90	81.81
3	17	Femenino	8	8	1	3	20	80	88.88
4	25	Masculino	7	7	3	3	20	70	70
5	62	Masculino	8	8	3	1	20	80	72.72
6	58	Femenino	8	8	3	1	20	80	72.72
7	28	Femenino	9	9	2	0	20	90	81.81
8	31	Masculino	9	9	1	1	20	90	90
9	16	Masculino	9	9	1	1	20	90	90
10	20	Femenino	8	8	3	1	20	80	72.72
11	46	Masculino	7	7	3	3	20	70	70
12	18	Femenino	9	9	0	2	20	90	100
13	22	Masculino	8	8	2	2	20	80	80
14	34	Masculino	7	7	3	3	20	70	70
15	16	Femenino	8	8	3	1	20	80	72.72

La Tabla 10 muestra los valores promedios de las pruebas realizadas.

Tabla 10

Resultados de valores promedio del sistema de detección de caídas en personas.

Valor promedio	Porcentaje
Exactitud Promedio	81.33%
Precisión promedio	78.67%
Verdaderos Positivos Promedio	40.67%
Verdaderos Negativos Promedio	40.67%
Falsos Positivos Promedio	11.33%
Falsos Negativos Promedio	7.33%

De igual forma se realiza una evaluación del sistema de detección de caídas mediante la herramienta Tensorboard. Tensorboard analiza internamente los datos de entrenamiento y toma una muestra del subconjunto test del set de datos. Desde la figura 71 hasta la figura 77 se muestran estos resultados.

Figura 71

*Evaluaciones de precisión y sensibilidad (Recall) con Tensorboard***.**

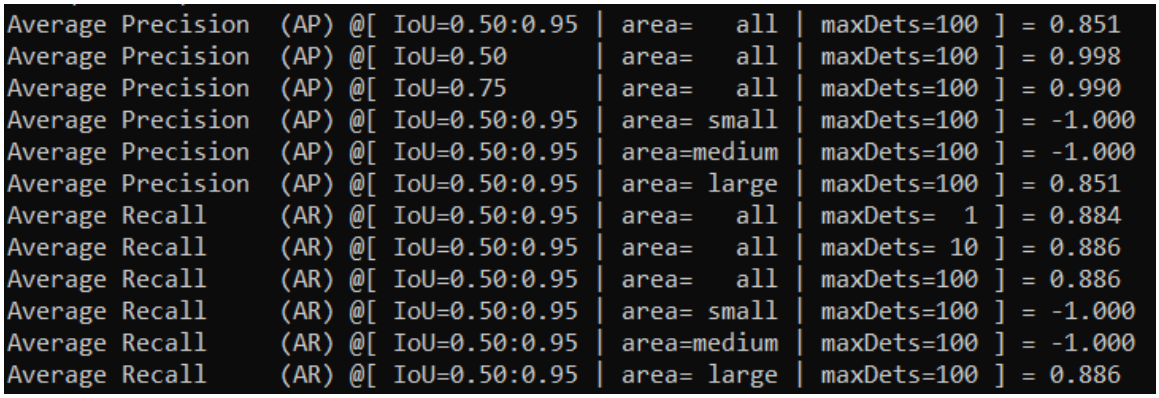


Figura 72

Evaluación de precisión, función de pérdida con Tensorboard

```
INFO:tensorflow:Eval metrics at step 45000
I0724 18:07:02.124587  8988 model_lib_v2.py:1015] Eval metrics at step 45000
INFO:tensorflow:         + DetectionBoxes_Precision/mAP: 0.851164
I0724 18:07:02.270765  8988 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP: 0.851164
INFO:tensorflow:         + DetectionBoxes_Precision/mAP@.50IOU: 0.997526
I0724 18:07:02.275793  8988 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP@.50IOU: 0.997526
INFO:tensorflow:         + DetectionBoxes_Precision/mAP@.75IOU: 0.990427
I0724 18:07:02.279794  8988 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP@.75IOU: 0.990427
INFO:tensorflow:         + DetectionBoxes_Precision/mAP (small): -1.000000
I0724 18:07:02.283784  8988 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP (small): -1.000000
INFO:tensorflow:         + DetectionBoxes_Precision/mAP (medium): -1.000000
I0724 18:07:02.285786  8988 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP (medium): -1.000000
INFO:tensorflow:         + DetectionBoxes_Precision/mAP (large): 0.851167
I0724 18:07:02.289782  8988 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP (large): 0.851167
INFO:tensorflow:         + DetectionBoxes_Recall/AR@1: 0.883914
I0724 18:07:02.293785  8988 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@1: 0.883914
INFO:tensorflow:         + DetectionBoxes_Recall/AR@10: 0.886321
I0724 18:07:02.298782  8988 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@10: 0.886321
INFO:tensorflow:         + DetectionBoxes_Recall/AR@100: 0.886412
I0724 18:07:02.302786  8988 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@100: 0.886412
INFO:tensorflow:         + DetectionBoxes_Recall/AR@100 (small): -1.000000
I0724 18:07:02.304795  8988 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@100 (small): -1.000000
INFO:tensorflow:         + DetectionBoxes_Recall/AR@100 (medium): -1.000000
I0724 18:07:02.308192  8988 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@100 (medium): -1.000000
INFO:tensorflow:         + DetectionBoxes_Recall/AR@100 (large): 0.886412
I0724 18:07:02.359424  8988 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@100 (large): 0.886412
INFO:tensorflow:         + Loss/localization_loss: 0.024146
I0724 18:07:02.363461  8988 model_lib_v2.py:1018]       + Loss/localization_loss: 0.024146
INFO:tensorflow:         + Loss/classification_loss: 0.094248
I0724 18:07:02.366456  8988 model_lib_v2.py:1018]       + Loss/classification_loss: 0.094248
INFO:tensorflow:         + Loss/regularization_loss: 0.079288
I0724 18:07:02.369211  8988 model_lib_v2.py:1018]       + Loss/regularization_loss: 0.079288
INFO:tensorflow:         + Loss/total_loss: 0.197683
I0724 18:07:02.372211  8988 model_lib_v2.py:1018]       + Loss/total_loss: 0.197683
INFO:tensorflow:Waiting for new checkpoint at Tensorflow\workspace\models\my_ssd_mobilenet_V2_FPNLite_320
```

Figura 73

Evaluación del modelo de CNN - Persona Sentada

Figura 74

Evaluación del modelo de CNN - Persona Caída

Figura 75

Evaluación del modelo de CNN - Persona Caída

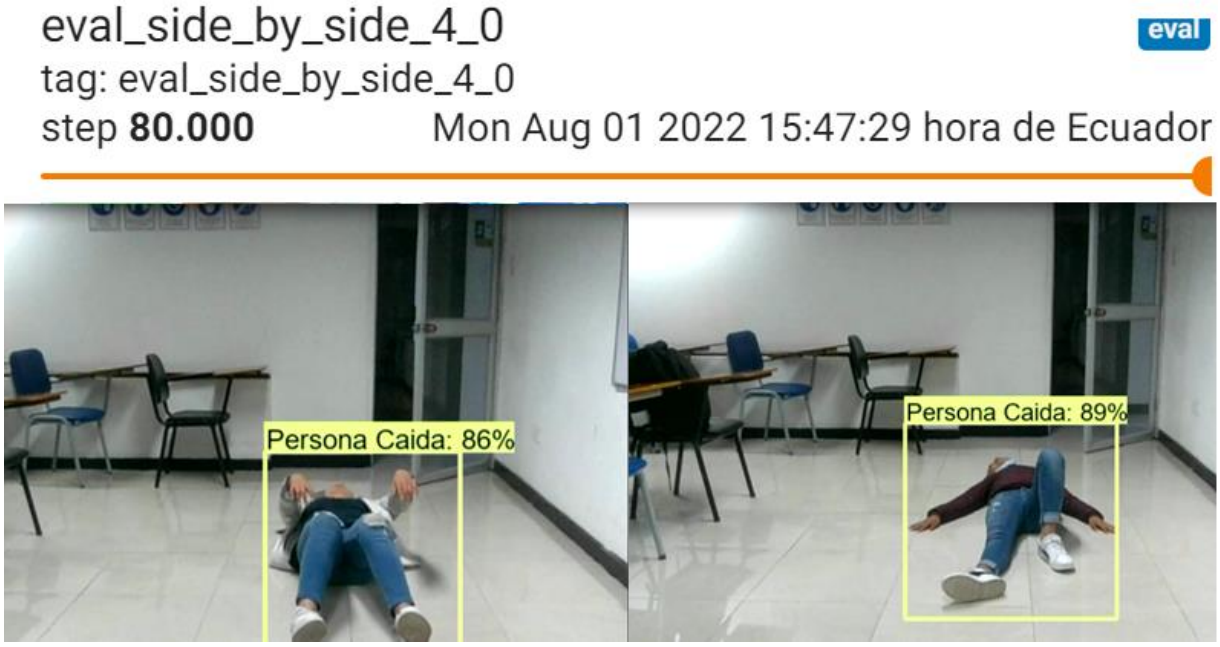


Figura 76

Evaluación del modelo de CNN - Persona Agachada

Figura 77

Evaluación del modelo de CNN - Persona Parada

Capítulo V. Conclusiones y recomendaciones

Conclusiones

- Se diseñó e implementó un sistema de detección de caídas en adultos mayores aplicando técnicas de aprendizaje profundo con una precisión superior al 80% utilizando para ello un modelo de redes neuronales convolucionales. Si bien esta precisión puede ser mejorada a través del incremento de set de datos, se decidió mantenerlo porque el objetivo es tener un sistema liviano que pueda ser implementado en un sistema embebido.
- Para lograr un buen funcionamiento del sistema fue esencial la creación de un set de datos con imágenes propias que facilitaron el aprendizaje del modelo de CNN, lo que permitió mejorar la eficiencia en las predicciones realizadas por la red neuronal convolucional.
- La elección del modelo SSD Mobilenet V2 como CNN permitió el ahorro de recursos computacionales comparado con otros modelos que tienen un mayor costo de memoria. La combinación de este modelo de CNN con el set de datos propios permitió tener un sistema liviano con buen nivel de predicción que podría ser evaluado para aplicaciones embebidas.
- La evaluación del sistema mediante herramientas de análisis de redes como Tensorboard permitió obtener la métrica función de pérdida en un valor, aproximado de 0.079 el cuál siendo cercano a cero se lo considera como aceptable.
- La utilidad del sistema de detección de caídas se evidencia ante la notificación de alerta mediante mensajería instantánea. Esto permite dar auxilio inmediato a la víctima reduciendo el impacto del accidente.

Recomendaciones

- Antes de iniciar con el proceso de desarrollo de un trabajo basado en detección de objetos. Es importante asegurar la compatibilidad entre las versiones de software y librerías de inteligencia artificial utilizadas. Para evitar inconvenientes que se pueden presentar en el entrenamiento o puesta en marcha del modelo de CNN.
- Cabe tomar en cuenta que las últimas versiones de software se encuentran generalmente en procesos de prueba, por lo que se sugiere seleccionar versiones anteriores de software que resultan más estables para el desarrollo de los proyectos. En alguna parte de este trabajo se presentó un problema relacionado con la utilización de la última versión de un software. Problema que fue solucionado al bajar las versiones de software.
- Como se indicó anteriormente, el set de datos es parte fundamental de la CNN. Motivo por el cual para mejorar el aprendizaje de la red neuronal se debe incrementar el número de imágenes. Por lo que se sugiere realizar un análisis costo beneficio del incremento de la precisión al incrementar el set de datos de las imágenes. Para el caso de este proyecto se debería incrementar las imágenes en cada instancia de clase: Persona Parada, Sentada, Agachada, acostada y caída.
- Si bien es cierto en la transferencia de aprendizaje se encuentran varios modelos de CNN pre – entrenados. Pero antes de seleccionar un modelo de trabajo, se deben analizar las métricas tamaño vs precisión. Tamaño referido a la memoria de almacenamiento y precisión al desempeño realizado en cuanto a las predicciones por clase.

Trabajos Futuros

- En el presente proyecto se usa la cámara del computador local para realizar la detección de caídas. En trabajos futuros se puede integrar este algoritmo con dispositivos móviles o cámaras de vídeo.
- En concordancia con los resultados obtenidos en el modelo SSD Mobilenet V2 se propone implementar nuevos detectores de caídas con otros modelos de CNN.
- El sistema de detección de caídas en personas desarrollado en este trabajo, puede ser incorporado como un sistema de videovigilancia. En lugares donde es preciso hacer un seguimiento a las personas.
- En trabajos futuros se pueden añadir funcionalidades interesantes, como conectar a bases de datos que permitan el registro diario de actividades de cada persona. Y también enviar notificaciones de alerta a dispositivos móviles o sistemas alojados en la web.

Bibliografía

Adhikari, Kripesh, Bouchachia, H., & Nait-Charif, H. (2017). *Fall Detection Dataset*. Obtenido de Activity recognition for indoor fall detection using convolutional neural network: https://falldataset.com/

Anaconda Inc. (20 de Marzo de 2022). Obtenido de https://www.anaconda.com/products/distribution

Arriola Oregui, I. (2018). Detección de objetos basada en Deep Learning y aplicada a vehículos autónomos. *Tesis de Máster*. Universidad del País Vasco.

Auvinet, E., Rougier, C., Meunier, J., St - Arnaud, A., & Rousseau, J. (Julio de 2010). *Conjunto de datos de caída de varias cámaras*. (Université de Montréal) Obtenido de http://www.iro.umontreal.ca/~labimage/Dataset/

Belshaw , M., Taati, B., Snoek, J., & Mihailidis, A. (2011). Hacia una solución pasiva de sensor único para la detección automática de caídas. *Sociedad de Ingeniería en Medicina y Biología del IEEE.*

Birodkar, V., Joglekar, S., & Rathod, V. (7 de mayo de 2021). *Github*. Obtenido de TensorFlow 2 Detection Model Zoo: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md

Brita Inteligencia artificial. (29 de octubre de 2021). Obtenido de Deep Learning Inteligencia Artificial Machine Learning: https://brita.mx/como-implementar-ia-para-resolver-tareas-de-procesamiento-de-imagenes#:~:text=Cuando%20se%20aplica%20al%20procesamiento,en%20im%C3%A1genes%20y%20videos%2C%20etc.

Calvo, D. (20 de Julio de 2017). *Red Neuronal Convolucional CNN.* Obtenido de https://www.diegocalvo.es/red-neuronal-convolucional/

Casilari, E., & SantoyoRamón, J. (2018). *Fall Detection Dataset (Universidad de Malaga).* Obtenido de https://figshare.com/articles/dataset/UMA_ADL_FALL_Dataset_zip/4214283

Castrillón Fernández, M., Bolaños, H., Enríquez, J., & Pencue-Fierro, L. (2007). SIstema asistido de visión artificial para la estandarización de los análisis de biopsias renales mediante microscopía óptica. *Facultad de CIencias de la Salud Universidad del Cauca*, 5.

Coder Solution. (31 de Mayo de 2022). Obtenido de https://coder-solution-es.com/solution-es-blog/1195260

Corona Ramírez, L., Abarca Jiménez, G., & Mares Carreño, J. (2014). *Sensores y actuadores. Aplicaciones con Arduino.* México D.F.: Grupo Editorial Patria.

de López Diz, S. (2019). Detección de acciones humanas a partir de información de profundidad mediante redes neuronales convolucionales. *Grado en Ingeniería Electrónica de Comunicaciones*. UNIVERSIDAD DE ALCALÁ.

De Luca, A., & Irigoitia, M. (2021). Análisis de la técnica Transfer Learning en Machine Learning a través de un caso de estudio: La clasificación de productos en el Banco Alimentario de La Plata. *Tesina de Licenciatura*. Universidad Nacional de La Plata.

Deng, J., Dong, W., Socher, R., Li, L.-J., & Fei-Fei, L. (2009). ImageNet: una base de datos de imágenes jerárquicas a gran escala. *Actas de la Conferencia IEEE de 2009 sobre visión por computadora y reconocimiento de patrones (CVPR) .* Miami, Estados Unidos.

Diaz Sola, M. (2015). *¿Cómo elegir un detector de caídas? - Fundación Alzheimer España*. Obtenido de http://www.alzfae.org/fundacion/461/como-elegir-un-detector-de-caidas

Flórez López, R., & Férnandez Férnandez, J. (2008). *Las redes neuronales artificiales. Fundamentos Teóricos y aplicaciones prácticas.* La Coruña: Netbiblo.

Freire, W. (2009 -2010). *Instituto Nacional de Estadística y Censos - INEC*. Obtenido de https://www.gerontologia.org/portal/archivosUpload/Ecuador-Encuesta-SABE-presentacion-resultados.pdf

Galeote Gutiérrez, D. (2018). Sistema de detección de caídas para hogares de personas mayores. Madrid: Universidad Politécnica de Madrid.

Gamboa Uribe, L. (26 de Noviembre de 2020). *Santander Global Technology & Operations*. Obtenido de Visión Artificial vs Visión Human: Así ven los ordenadores: https://santandergto.com/vision-artificial-vs-vision-humana-asi-ven-los-ordenadores/

Gámez Albán, H. (2016). Estructura general de una red neuronal artificial. *EIA*.

García Crespo, Á., Rodríguez Goncalvez, R., Garcés, A., & García Tejedor, Á. (s.f.). *Centro Internacional sobre el Envejecimiento - CENIE*. Recuperado el 4 de Agosto de 2021, de https://cenie.eu/es/blogs/securhome/deteccion-automatica-de-actividad-en-el-hogar-de-personas-mayores-para-su-asistencia

Giménez-Palomare, F., Monsoriu, J., & Alemany-Martínez, E. (2016). Aplicación de la convolución de matrices al filtrado de imágenes. *Modelling in Science Education and Learning*.

GitHub. (24 de Marzo de 2022). Obtenido de protobuf: https://github.com/protocolbuffers/protobuf/releases

Glorot, X., Bordes, A., & Bengio, Y. (2011). *Deep Sparse Rectifier Neural Networks.* Obtenido de Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics: http://proceedings.mlr.press/v15/glorot11a.html

González Díaz, S. (2017). Sistema de detección de caídas de personas mayores por medio de visión artificial. Universidad Carlos III de Madrid.

Guru99. (3 de Julio de 2022). Obtenido de Tutorial de Tensorboard: Visualización de gráficos con ejemplo: https://guru99.es/tensorboard-tutorial/

Izama Flores, K. (2017). “SISTEMA ELECTRÓNICO DE ALARMA DE CAÍDAS PARA ADULTOS MAYORES DEL CENTRO DE CUIDADO DEL ADULTO MAYOR SAN MARTÍN”. Ibarra: Universidad Técnica del Norte.

Jiménez, P. (2009). Estudio, diseño y optimización de técnicas de visión artificial para su aplicación a los sistemas de videovigilancia". Alcalá de Henares: Universidad de Alcalá.

Kępski, M., & Kwolek, B. (Diciembre de 2014). *UR Fall Detection Dataset*. Obtenido de Human fall detection on embedded platform using depth maps and wireless accelerometer, Computer Methods and Programs in Biomedicine: http://fenix.univ.rzeszow.pl/~mkepski/ds/uf.html

Khandelwal, R. (30 de noviembre de 2019). *Towards Data Science*. Obtenido de SSD : Single Shot Detector for object detection using MultiBox: https://towardsdatascience.com/ssd-single-shot-detector-for-object-detection-using-multibox-1818603644ca#:~:text=Single%20Shot%20detector%20like%20YOLO,object%20detection%20models%20on%20VOC2007

Lin, T.-Y., Patterson, G., Ronchi, M., Cui , Y., Maire, M., Belongie, S., . . . Dollár, P. (2017). *COCO*. Obtenido de Common Objects in Context: https://cocodataset.org/#home

Liu, Z., Cao, Y., Cui, L., Song, J., & Zhao, G. (2018). Una base de datos de referencia y una evaluación de referencia para la detección de caídas basada en sensores portátiles para la plataforma Internet of Medical Things. IEEE.

MathWorks. (12 de junio de 2022). Obtenido de Transfer Learning: https://la.mathworks.com/discovery/transfer-learning.html

Matich, D. (Marzo de 2001). Redes Neuronales: Conceptos Básicos y aplicaciones. *Informática Aplicada a la Ingeniería de Procesos – Orientación I*. Rosario: Universidad Tecnológica Nacional – Facultad Regional Rosario.

Medina Muñoz, L. A., González-López, S., Mayorquín Robles, J., & López Valencia, G. (2019). Diseño de interfaces de inspección de calidad utilizando redes neuronales y visión artificial para la industria de manufactura. *Research in Computing Science 148(8)*, 121-131.

Ministerio de Educación. (2012). *Aplicación práctica de la visión artificial en el control de procesos industriales.* Gobierno de España.

NetApp. (5 de Junio de 2022). Obtenido de ¿Qué es la inteligencia artificial?: https://www.netapp.com/es/artificial-intelligence/what-is-artificial-intelligence/

Núñez-Marcos, A., Azkune, G., & Arganda-Carreras, I. (2017). Vision-Based Fall Detection with Convolutional Neural Networks. *Wireless Communications and Mobile Computing*, 16.

Nvidia Developer. (12 de Marzo de 2022). Obtenido de https://developer.nvidia.com/cuda-toolkit-archive

Nvidia Developer. (12 de Marzo de 2022). Obtenido de https://developer.nvidia.com/rdp/cudnn-archive

Olabe, X. (1998). Redes neuronales artificiales y sus aplicaciones. *Dpto. Ingeniería de sistemas y automática*. Escuela Superior de Ingeniería de Bilbao, EHU.

Oppenhein, A., & Schafer, R. (2010). *Discrete-Time Signal Processing.* Prentice Hall.

Organización Mundial de la Salud. (26 de Abril de 2021). Obtenido de https://www.who.int/es/news-room/fact-sheets/detail/falls

Osimani, C., & Kohmann, E. (2015). Simple método para interacción natural Humano - Computadora con movimientos del rostro. *Centro de investigación aplicada y desarrollo en Informática y Telecomunicaciones*.

Pandey, A. (2018). Depth-wise Convolution and Depth-wise Separable Convolution. *Medium*.

Pelegrí, J. (22 de Mayo de 2019). *Universal Robots*. Obtenido de Ventajas y aplicaciones de visión artificial en robots colaborativos: https://www.universal-robots.com/es/blog/vision-artificial-en-robots/

Percat Fedalto, F. (2015). Producto de Convolución: Aplicaciones. *Universidad Nacional del Sur*.

Pérez Lorenzo, C. (Junio de 2019). Detección precoz de cáncer de piel en imágenes basado en redes convolucionales. Universidad Autónoma de Madrid.

Pérolle, G., & Etxeberria Arritxabal, I. (2017). Detector automático de caídas y monitorización de actividad para personas mayores. *Revista Española de Geriatría y Gerontología*.

Pose, M. (2009). Introducción a las redes de neuronas artificiales. *Departamento de Tecnologías de la Información y las Comunicaciones*. Universidad da Coruña. Obtenido de Departamento. Tecnologías de la información y las comunicaciones. Universidad da Coruña: http://sabia.tic.udc.es/mgestal/cv/RNAtutorial/TutorialRNA.pdf

QoChuk, B. (12 de junio de 2022). *OpenGenus IQ: Computing Expertise & Legacy*. Obtenido de Single Shot Detector (SSD) + Architecture of SSD: https://iq.opengenus.org/single-shot-detector/

Ramalingam, B., Mohan, R., Balakrishnan, S., Elangovan, K., Gómez, B., Pathmakumar, T., . . . Baskar, C. (2021). sTetro- Deep Learning Powered Staircase Cleaning and Maintenance Reconfigurable Robot. *Sensors*, 17. Obtenido de sTetro- Deep Learning Powered Staircase Cleaning and Maintenance Reconfigurable Robot.

Rodríguez, V. (30 de octubre de 2018). *Conceptos básicos sobre redes neuronales*. Obtenido de https://vincentblog.xyz/posts/conceptos-basicos-sobre-redes-neuronales

Sanabria S, J., & Archila D, J. (2011). Detección y análisis de movimiento usando visión. *Redalyc*, 10.

Shah, K. (7 de diciembre de 2019). *Towards Tech Intelligence*. Obtenido de A Quick Overview to the Transfer Learning and it's Significance in Real World Applications: https://medium.com/towards-tech-intelligence/a-quick-overview-to-the-transfer-learning-and-its-significance-in-real-world-applications-790fb57debad

Sherin P, S. (2019). Human Fall Detection using Convolutional Neural Network. *INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 08*.

Solución Ingenieril. (11 de Mayo de 2022). Obtenido de http://solucioningenieril.com/vision_artificial/etapas_de_un_sistema_de_vision#:~:text=Escenario%20a%20analizar%3A%20Es%20el,unidad%20donde%20pueda%20ser%20procesada.

Tsang, S.-H. (2019). Review: MobileNetV2 — Light Weight Model (Image Classification). *Towards Data Science*.

Tzutalin. (2015). *LabelImg*. Obtenido de Git code: https://github.com/tzutalin/labelImg

Vázquez Rull, M. (2016). Reconocimiento de Objetos usando Deep Learning. *Dep. de Ingeniería de Sistemas y Automática*. Sevilla: Universidad de Sevilla.

Vladimirov, L. (24 de Marzo de 2022). *Tutorial de la API de detección de objetos de TensorFlow*. Obtenido de https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/install.html

Xeridia. (16 de septiembre de 2019). Obtenido de Redes Neuronales artificiales: Qué son y cómo se entrenan: https://www.xeridia.com/blog/redes-neuronales-artificiales-que-son-y-como-se-entrenan-parte-i

Yu, H., Chen , C., Du, X., Li, Y., Rashwan, A., Hou, L., . . . Li, J. (24 de Marzo de 2020). *TensorFlow Model Garden*. Obtenido de https://github.com/tensorflow/models

Zúñiga-López, A., Avilés-Cruz, C., Ferreyra-Ramírez, A., & Rodríguez-Martínez, E. (2020). Algoritmo de clasificación basado en la función Softmax. *Research in Computing Science*. México: Universidad Autónoma Metropolitana.

Apéndice