

**Construcción de un conjunto de datos robusto para la detección de caídas
basado en secuencias de imágenes RGB y de profundidad**



Jesús Esteban Fajardo Muñoz

Corporación Universitaria ComfacaUCA - UnicomfacaUCA

Facultad de Ingenierías

Ingeniería de Sistemas

Popayán, Colombia

2021

**Construcción de un conjunto de datos robusto para la detección de caídas
basado en secuencias de imágenes RGB y de profundidad**



Jesús Esteban Fajardo Muñoz

Proyecto de grado para optar al título de Ingeniero de Sistemas

Director: MSc. Julián Fernando Muñoz Ordóñez

Codirector: Ing. Cristian Dubán Chilito

Corporación Universitaria ComfacaUCA - UnicomfacaUCA

Facultad de Ingenierías

Ingeniería de Sistemas

Popayán, Colombia

2021

DATOS GENERALES	
Facultad:	Ingeniería
Programa académico:	Ingeniería de Sistemas
Título del proyecto:	Construcción de un conjunto de datos robusto para la detección de caídas basado en imágenes RGB y de profundidad.
Grupo de investigación:	TIC Unicomfauca
Línea de investigación:	Inteligencia Artificial & Big Data
Proyecto institucional asociado:	Vigilancia de pacientes para prevenir eventos adversos relacionados con caídas en ambientes hospitalarios utilizando inteligencia artificial.
Fecha de aprobación del perfil:	
Fecha de elaboración de la propuesta:	15/11/2020
Fecha de radicación:	

Nota de aceptación

Aprobado por el comité de grado en cumplimiento de los requisitos exigidos por la
Corporación Universitaria Comfacaucá - Unicomfacaucá para optar al título de Ingeniero
de Sistemas.

Ingeniero

Ingeniero

CONTENIDO

CAPÍTULO 1 – BASE DEL PROYECTO	1
1.1. PLANTEAMIENTO PROBLEMA	1
1.2. JUSTIFICACIÓN	4
1.3. OBJETIVOS	6
1.3.1. General	6
1.3.2. Específicos	6
1.4. ESTADO DEL ARTE.....	7
CAPÍTULO 2 - EXPLICACIÓN	16
MARCO TEÓRICO.....	16
2.1. Caídas.....	16
2.2. Dataset.....	17
2.2.1. Common Objects in Context (COCO)	19
2.2.2. ImageNet.....	20
2.3. Deep learning	20
2.4. Redes neuronales convolucionales.....	21
2.5. Capa convolucional.....	21
2.6. Capa pooling	21
2.7. Capa fully-connected (FC).....	22
2.8. Capa dropout	22
2.9. Funciones de activación	22
2.10. Arquitecturas de red neuronal usadas en el proyecto	22
2.10.1. VGG16.....	22
2.10.2. ResNet.....	25
2.10.3. InceptionV3.....	26
2.10.4. Xception	28
2.10.5. DenseNet.....	29
2.10.6. EfficientNetB7	30
2.10.7. MobileNetV2	32
2.11. Métricas de clasificación.....	33

CAPÍTULO 3 – DESARROLLO DEL PROYECTO.....	36
3. METODOLOGÍA	36
3.2. Fase II.....	39
3.4. Fase IV	48
3.5. Fase V	53
CAPÍTULO 4 - ALGORITMO	58
4.1. ALGORITMO DE RED NEURONAL	58
CAPÍTULO 5 - RESULTADOS.....	72
5. RESULTADOS.....	72
5.1. ENTRENAMIENTO	72
5.2. VALIDACIÓN	77
CAPÍTULO 6.....	79
CONCLUSIONES	79
REFERENCIAS.....	82

LISTA DE FIGURAS

Figura 1. Ejemplo de data estructurada.....	18
Figura 2. Detección de objetos.....	19
Figura 3. ImageNet Dataset	20
Figura 4. Modelo de red neuronal VGG16	24
Figura 5. Modelo de red neuronal ResNet	25
Figura 6. Modelo de red neuronal Inception.....	27
Figura 7. Capas del modelo de red neuronal Xception	28
Figura 8. Modelo de red neuronal DenseNet	30
Figura 9. Capas del modelo de red neuronal Efficient Net	31
Figura 10. Modelo de red neuronal MobileNetV2.....	32
Figura 11. Capas del modelo de red neuronal MobileNetV2.....	32
Figura 12. Diagrama de comparación exactitud y precisión.....	34
Figura 13. Matriz de confusión, True Positive - TP (Verdadero Positivo): Predicción positiva y valor real positivo. True Negative - TN (Verdadero Negativo): Predicción negativa y valor real negativo. False Positive - FP (Falso Positivo): Predicción positiva y valor real negativo. False Negative - FN (Falso Negativo): Predicción negativa y valor real positivo. Fuente: bookdown.	35
Figura 14. Metodología CRISP-DM [61], [62].....	36
Figura 15. Arquitecturas de redes neuronales convolucionales aplicadas en la resolución del conjunto de datos ImageNet. Fuente: paperswithcode [63].....	37
Figura 16. Precisión de clasificación de AlexNet, VGG-16, ResNet-152, Inception y Xception en ImageNet.....	38
Figura 17. Ejemplo de fragmento vacío.....	40
Figura 18. Ejemplo de fragmento muerto [más de 5 segundos en la misma postura después de realizar una acción].....	40
Figura 19. Ejemplo de acciones múltiples en un mismo video [secuencia extraída de 13 segundos de vídeo, donde al principio y final del mismo existen fragmentos muertos, además]	40
Figura 20. Vídeos del BOSS Dataset Parte 1.....	42
Figura 21. Vídeos del BOSS Dataset Parte 2.....	43
Figura 22. Vídeos del BOSS Dataset Parte 3.....	44

Figura 23. Vídeos del BOSS Dataset Parte 4.....	44
Figura 24. Vídeos del BOSS Dataset Parte 5.....	45
Figura 25. Vídeos del BOSS Dataset Parte 6.....	45
Figura 26. Tiempo de descarga desde el servidor que está alojado el BOSS Dataset	46
Figura 27. Optimización de tamaño BOSS Dataset	47
Figura 28. Peso de los archivos resultantes del BOSS Dataset después de recorte y depuración.....	48
Figura 29. Frames extraídos del primer video del dataset – Clase ADL	49
Figura 30. Frames extraídos del 27vo video del dataset – Clase ADL	49
Figura 31. Cantidad de frames extraídos de todos los vídeos	49
Figura 32. CSV Frames generados	50
Figura 33. Cantidad de vídeos en el RFDD	50
Figura 34. CSV Ajustado a la cantidad de vídeos del RFDD	50
Figura 35. Pesos de arquitecturas de red después del entrenamiento	51
Figura 36. Entrenamiento de arquitecturas de red con RFDD	52
Figura 37. Carga de pesos con arquitectura de red neuronal correspondiente.....	53
Figura 38. Ejemplos de las diferentes capas finales de arquitecturas de red	53
Figura 39. Tiempos de validación de arquitecturas de redes neuronales convolucionales	54
Figura 40. Archivo de entrenamiento (trainlist01.txt)	58
Figura 41. Archivo de prueba (testlist01.txt)	58
Figura 42. Árbol de directorios para el proyecto	58
Figura 43. Lectura de la lista de videos en los archivos de entrenamiento y validación	59
Figura 44. Creación de etiquetas para vídeos de entrenamiento y prueba	60
Figura 45. Nombres de video con su respectiva etiqueta.....	60
Figura 46. Extracción de muestras del dataset	60
Figura 47. Proceso de extracción de muestras completado correctamente	61
Figura 48. Creación del archivo csv.....	62
Figura 49. Lectura de csv y agregación de imágenes a la lista de entrenamiento.....	64
Figura 50. Entrenamiento parte #1.....	65
Figura 51. Entrenamiento parte #2.....	65
Figura 52. Entrenamiento parte #3.....	66
Figura 53. Carga de pesos y definición de arquitectura de red	67

Figura 54. Resumen de la arquitectura de red neuronal especificada (Figura de VGG16).....	68
Figura 55. Especificación de nombre de capa de la arquitectura de red, realización del producto de sus pesos y definición de modelo de arquitectura	69
Figura 56. Carga de pesos, obtención de lista de validación, creación de dataframe, etiquetas y listas de almacenamiento	69
Figura 57. Bucle de procesamiento de muestras de vídeo	70
Figura 58. Procesamiento de frames y predicción de la data.....	71
Figura 59. Métricas de evaluación: pérdida y precisión para la arquitectura VGG16 resolviendo el conjunto de datos RFDD.....	73
Figura 60. Métricas de evaluación: pérdida y precisión para la arquitectura ResNet50 resolviendo el conjunto de datos RFDD.....	73
Figura 61. Métricas de evaluación: pérdida y precisión para la arquitectura InceptionV3 resolviendo el conjunto de datos RFDD.....	74
Figura 62. Métricas de evaluación: pérdida y precisión para la arquitectura Xception resolviendo el conjunto de datos RFDD.....	74
Figura 63. Métricas de evaluación: pérdida y precisión para la arquitectura DenseNet121 resolviendo el conjunto de datos RFDD.....	75
Figura 64. Métricas de evaluación: pérdida y precisión para la arquitectura ResNet101 resolviendo el conjunto de datos RFDD.....	75
Figura 65. Métricas de evaluación: pérdida y precisión para la arquitectura EfficientNetB7 resolviendo el conjunto de datos RFDD.....	76
Figura 66. Métricas de evaluación: pérdida y precisión para la arquitectura MobileNetV2 resolviendo el conjunto de datos RFDD.....	76
Figura 67. Matriz de confusión VGG16	77
Figura 68. Matriz de confusión ResNet50	77
Figura 69. Matriz de confusión InceptionV3	78
Figura 70. Matriz de confusión Xception	78
Figura 71. Matriz de confusión DenseNet121	78
Figura 72. Matriz de confusión ResNet101	78
Figura 73. Matriz de confusión EfficientNetB7.....	78
Figura 74. Matriz de confusión MobileNetV2.....	78

LISTA DE TABLAS

Tabla 1. Datasets públicos usados en el proyecto	56
Tabla 2. Resultados de entrenamiento con diferentes arquitecturas de red neuronal.....	72
Tabla 3. Resultados de validación con diferentes arquitecturas de red neuronal.....	77

RESUMEN

Las caídas suponen un gran problema contra la salud de los pacientes a nivel mundial, siendo la segunda causa de muerte accidental, según la Organización Mundial de la Salud –OMS–, creando, de esta forma, un foco de investigación para muchos, el cual busca disminuir el porcentaje de aquellos eventos de caídas que llegan a tornarse mortales tanto en ambientes controlados como no controlados y en población de cualquier edad; sin embargo, se sabe que la población más vulnerable a sufrir heridas de gravedad es aquella en la que los individuos se encuentran en la tercera edad.

Por esta razón, se propone un conjunto de datos llamado *Robust Fall Detection Dataset* –RFDD– compuesto de una amplia variedad de acciones del diario vivir (ADL), como caminar, sentarse, levantarse, entre otros, y acciones de caídas simuladas por actores reales y en diferentes escenarios. Después de una validación del RFDD, usando un algoritmo de red neuronal y diferentes arquitecturas de redes neuronales, se obtienen resultados bastante positivos, esto con el objetivo de ayudar en investigaciones futuras por parte de cualquier equipo de investigación al proporcionar un conjunto de datos bastante completo y de acceso gratuito.

ABSTRACT

Falls are a great problem against the health of patients worldwide, being these the second cause of accidental death according to the World Health Organization –WHO– thus creating a focus of research for many, seeking to reduce the percentage of Those fall events that become fatal both in controlled and uncontrolled environments and in the population of any age, however, it is known that the population most vulnerable to suffering serious injuries are those individuals who are in the elderly.

A Dataset called Robust Fall Detection Dataset –RFDD– is proposed, composed of a wide variety of daily living actions (ADL) such as walking, sitting, getting up, among others, and falling actions, simulated by real actors and in different scenarios. After a validation of the RFDD using a neural network algorithm and different neural network architectures, quite positive results are obtained, this with the aim of helping in future research by any research team by providing a fairly complete data set and free access.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 1 de 104

CAPÍTULO 1 – BASE DEL PROYECTO

1.1. PLANTEAMIENTO PROBLEMA

Según la Organización Mundial de la Salud –OMS–, las caídas son la segunda causa de muerte en el mundo por lesiones accidentales o no intencionales. Las cifras de muertes al año superan las 600 000 personas, siendo los hombres quienes sufren mayor índice de caídas mortales, en comparación con las mujeres. Más del 70 % de estos accidentes ocurren en países en vía de desarrollo, donde los recursos son escasos y las personas de edad mayores a 65 años son las más afectadas [1]. Lo anterior se debe a que, a medida en que la edad de las personas aumenta, el cuerpo adquiere una mayor vulnerabilidad a los golpes. Las caídas causan traumatismos que en personas de avanzada edad pueden llegar a desencadenar la muerte. Entre los años 2017 y 2019, se observó un crecimiento en la población anciana a nivel mundial, incrementando de un 8,6 % a un 9 % [2]. Además, se estima que para el año 2050 estas cifras incrementarán a más del 20 % [3], lo que constata que este crecimiento de la población de avanzada edad deberá ser considerado en el manejo de los eventos adversos causados por las caídas.

Las instituciones y entidades prestadoras de salud, como hospitales, clínicas y ancianatos encargadas del cuidado de los pacientes, tienen a su disposición un número determinado de talento humano responsable de la vigilancia y del bienestar del paciente. En este sentido, por lo general, siempre será menor la cantidad de personal de la salud frente a la cantidad de pacientes que están presentes en los sitios de atención, lo que genera una gran diferencia que dificulta la vigilancia continua. Los hospitales, que son instituciones que albergan una población abundante de pacientes (ambulatorios y hospitalizados) no son ajenos a la problemática de las caídas. En diferentes estudios recolectados a nivel mundial se estima que únicamente en Estados Unidos, las caídas de pacientes tienen un costo para el Gobierno de 3,4 millones de dólares en el 2020, y que para el

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 2 de 104

2030, la cifra aumentará a 5,7 millones de dólares [4]. El departamento de medicina legal en Colombia, en el año 2018, reportó que las personas muertas por causa de una caída ascendían a 3523, de las cuales 1106 casos han sido ocasionados desde su propia altura, lo que se considera una tasa bastante alta de afectaciones [5].

Aunque los circuitos cerrados de televisión –CCTV– instalados en los hospitales proporcionan un grado de vigilancia a los pacientes, su principal objetivo no es la detección y alertas tempranas relacionadas con caídas. Diversas investigaciones proponen sistemas de detección de caídas en ambientes controlados que van desde dispositivos usables [6] a cámaras de profundidad, tales como: Kinect [7], [8]. La mayor parte de estas investigaciones están enfocadas en personas de la tercera edad, puesto que dicha población es altamente vulnerable a las caídas, debido a su condición física, tal como se mencionó anteriormente. En los dispositivos usables, la principal desventaja radica en que son incómodos e invasivos, razón por la cual las personas tienden a olvidarlos o no quieren usarlos [9], generando así que en caso de una caída sea imposible alertar al personal encargado, impidiendo prestar una atención inmediata y oportuna. Los sensores de profundidad son dispositivos no invasivos que facilitan la protección de datos, relacionada con la privacidad del paciente, debido a que trabajan con longitudes de onda del infrarrojo, por lo que no es posible el reconocimiento de un rostro humano. Sin embargo, diversas investigaciones solo utilizan los puntos claves (simplificación de las imágenes de profundidad) obtenidos por los SDK, como Microsoft Kinect, reduciendo el cuerpo humano a solo un conjunto de puntos [9], [10]. Después, mediante métodos clásicos de clasificación, o basados en características, realizan el rastreo del esqueleto humano y predicen diferentes eventos relacionados con caídas; sin embargo, este enfoque falla cuando las condiciones lumínicas cambian bruscamente, evitando la correcta calibración del sensor y produciendo malas ubicaciones de los puntos sobre las articulaciones de la persona.

La gran mayoría de los sistemas de detección de caídas son entrenados con conjuntos de datos propios creados a través de un ambiente estructurado y controlado. Por lo general, el sistema no es

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 3 de 104

entrenado ni probado en diferentes ambientes, por lo que los conjuntos de datos no tienen una diversificación de los escenarios. Aunque son diversos los trabajos realizados en el tema, muy pocos de los sistemas son implementados en ambientes reales o se convierten en productos finales [11]; además, se suma que por los altos costos, Colombia, al ser un país en vía de desarrollo, no da prioridad a las investigaciones de detección de caídas causando así, la falta de recursos para suplir un proyecto de estos, llevando a que las campañas de prevención de caídas sean llevadas a cabo en lugar de investigaciones [5], [12]. Sin embargo, no se puede desconocer que aunar la tecnología, la ciencia y los esfuerzos del personal médico generaría notables mejoras a los problemas de salud pública.

Por otra parte, implementar un sistema que esté enfocado al cuidado de la salud siempre debe tener un rigor científico y técnico que brinde la confianza para funcionar de manera adecuada en un ambiente real; hasta el momento, no existe un conjunto de datos estandarizado que pueda usarse en la detección de caídas de pacientes, en otras palabras, un conjunto de datos que abarque gran cantidad de muestras relacionadas con caídas y actividades del diario vivir en diferentes escenarios y que tenga un carácter de acceso libre que permita el fortalecimiento de nuevos o mejorados sistemas para la detección de caídas.

En este trabajo se plantea la siguiente pregunta de investigación: ¿cómo construir y estandarizar un conjunto de datos relacionado con eventos de caídas y actividades del diario vivir que favorezca la creación de sistemas de detección de caídas más robustos para el funcionamiento en ambientes reales?

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 4 de 104

1.2. JUSTIFICACIÓN

Los proyectos relacionados con la detección o predicción de caídas precisan de un conjunto de datos para funcionar. **El presente proyecto busca dar solución a la carencia de un conjunto de datos robusto** y con una diversificación de las actividades realizadas por las personas y los escenarios con diferentes ambientes. En la actualidad, existe una amplia gama de conjuntos de datos para la detección de caídas; sin embargo, la mayoría carecen de diversificación de escenarios cuando las pruebas son realizadas en un único laboratorio, así como la diversificación de eventos, cuando son pocos los actores que realizan las acciones, y la falta de estandarización de las muestras, debido a que cada autor maneja diferentes tipos de almacenamiento: imágenes, videos, texto, etc. Lo anterior es fácilmente explicable, puesto que construir un conjunto de datos robusto requiere un esfuerzo por parte del equipo de investigación, invirtiendo gran cantidad de tiempo en la creación de este conjunto de muestras, al igual que los actores que participan en dicha creación, quienes deben esforzarse en recrear con gran precisión una caída para garantizar la calidad de las muestras del set de datos.

En la mayor parte de los casos, respecto a los proyectos relacionados con la detección de caídas, cuando el equipo no puede costear un sensor de profundidad y elaborar su propio conjunto de datos, se usa uno existente que sea público. En este orden de ideas, los investigadores, cuyos actuales o futuros proyectos estén relacionados con el tema de detección y/o predicción de caídas, son quienes se ven beneficiados al tener a disposición un conjunto de datos público y diverso en actividades, con una amplia cantidad de muestras que pueda asemejarse a un ambiente real, dando al investigador la posibilidad de tener más oportunidades de obtener mejores resultados. Si bien muchos de estos se componen de una muy amplia cantidad de secuencias de imágenes, resultando en conjuntos de datos de gran tamaño, no todos son de buena calidad.

Al crear un conjunto de datos cuyo objetivo sea mejorar la variedad de ambientes, las actividades del diario vivir, la simulación de caídas, la variabilidad en la posición de la cámara y

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 5 de 104

la disposición de diferentes actores con características físicas distintas, se realiza un avance en la problemática de caídas de pacientes, el cual en los últimos años se ha convertido en un tema importante y crítico, ya que, según la OMS, las caídas son la segunda causa de muerte en el mundo provocada por lesiones accidentales o no intencionales.

El conjunto de datos estará disponible de manera pública y contará con una licencia que apoyará la realización de proyectos. Este podrá ser descargado desde un servidor de almacenamiento confiable que, además de ser seguro, no limita el ancho de banda en la descarga, como sí lo hacen otros servidores. Puesto que el tiempo es algo importante en cualquier investigación, para este caso específico de proyectos de detección y/o predicción de caídas, muchos conjuntos de datos son alojados en servidores que limitan el ancho de banda, volviéndose tediosos a la hora de la descarga y consumiendo una gran cantidad de tiempo, provocando retrasos considerables en el proceso investigativo.

Actualmente, las redes neuronales profundas han marcado la tendencia en la solución de los problemas de caídas de pacientes; sin embargo, es bien conocido que estos algoritmos requieren de un conjunto de datos amplio y representativo para lograr excelentes resultados. Es por eso que, gracias a la variedad de muestras y de ambientes que conforman el conjunto de datos planeado en esta investigación, podrá ser emulado en gran medida el comportamiento humano relacionado a las actividades del diario vivir y las caídas, mejorando así la vigilancia en pacientes. De esta manera, se puede apuntar a lograr el objetivo principal de muchas investigaciones: realizar una detección y/o predicción de caídas confiable y acertada.

Finalmente, se busca incentivar a la comunidad interesada en solucionar las caídas de pacientes, realizando un aporte de conocimiento, con el fin de proporcionar más herramientas en la búsqueda de lograr reducir el porcentaje de muertes por caídas accidentales, ayudando a crear sistemas de detección y/o predicción de caídas mediante el ahorro de tiempo y recursos que conllevan la creación de un conjunto de datos.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 6 de 104

1.3. OBJETIVOS

1.3.1. General

Construir un conjunto de datos robusto compuesto por secuencias de imágenes basadas en eventos adversos de caídas de personas y actividades del diario vivir, con el fin de apoyar investigaciones relacionadas a la seguridad del paciente.

1.3.2. Específicos

- Establecer el estado del arte de los conjuntos de datos relacionados con la detección de caídas, utilizados en diversos proyectos de seguridad del paciente.
- Definir un algoritmo que permita estandarizar y consolidar conjuntos de datos validados en el estado del arte en un conjunto de datos robusto relacionado con la detección de caídas, que se caracterice por su diversidad de acciones y escenarios.
- Validar el conjunto de datos construido a través de un algoritmo de red neuronal convolucional profunda basado en sus métricas de precisión.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 7 de 104

1.4. ESTADO DEL ARTE

Una caída, dependiendo del tipo (a igual nivel o desde una elevación), llega a generar problemas de salud delicados para la persona que las sufre y de ciertos factores como la altura, la distancia, la velocidad, entre otros. Se entiende que una caída es una actividad anormal, ya que ocurre por factores externos y no es una acción intencional por parte de un individuo. La mayoría de caídas ocurren por falta de balance en la persona, es decir, la pérdida del equilibrio causada por diversos tipos de movimientos como los ejercicios, el agacharse, los resbalones, etc. La población más afectada por estos sucesos son los ancianos, al ser más vulnerables por su edad que el resto, debido a que cuentan con un estado físico más deteriorado con respecto al de los individuos más jóvenes, esto conlleva, en la mayoría de los casos, a la muerte de la persona. [13]

Diferentes investigaciones relacionadas con las caídas de pacientes han sido trabajadas durante los últimos años. Algunas se enfocan en la aplicación de algoritmos de aprendizaje-máquina o técnicas de visión por computadora para resolver adecuadamente un conjunto de datos relacionados con actividades del diario vivir o caídas. En la mayoría de los proyectos, un conjunto de datos es creado por parte de los investigadores. A continuación, se presenta el estado del arte relacionado con la creación de nuevos conjuntos de datos para la detección de caídas en diferentes ambientes.

Existen muchos sistemas de detección de caídas comerciales que usan datos provenientes de acelerómetros y/o giroscopios sujetos al cuerpo de las personas capturando las señales para detectar la ocurrencia de un evento de caída. Además, estos sistemas usan sensores que están vinculados con dispositivos móviles que contienen funcionalidades como las llamadas a servicios de emergencias en caso de ser requerido. En el año 2013, George Yavoulas *et al.* [14] crean un conjunto de datos que ayuda en la prueba de nuevos métodos o la comparación de algoritmos existentes para la detección de caídas y el reconocimiento de actividades diarias. El dataset contiene señales registradas con el acelerómetro y el giroscopio. El conjunto de datos en cuestión

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 8 de 104

sirve para ser usado como *benchmark* en futuros algoritmos, con el fin de ahorrar tiempo y recursos para los investigadores interesados en la problemática de las caídas de pacientes en el mundo.

Cada algoritmo tiene sus fortalezas y sus debilidades, no todos funcionan de la misma forma, aunque traten con el mismo conjunto de datos, es por ello que los autores no publican el conjunto de datos como algo definitivo, sino como una ayuda en el autoaprendizaje de los sistemas que lo implementen. De todas maneras, el principal problema que tiene este dataset es la corta duración de los datos restantes después del final de una determinada actividad realizada por un actor. A futuro, los investigadores proponen robustecer el conjunto de datos con más sujetos que realicen las actividades y también una evaluación de los algoritmos de detección de caídas y el reconocimiento de actividades diferentes a las caídas para evitar falsas alarmas.

En 2014, Bogdan Kwolek *et al.* [15] explican que la principal razón de la no aceptación a los métodos disponibles actualmente para ancianos es la gran cantidad de falsas alarmas que generan los sensores en los dispositivos que se usan para detectar las caídas, ya que muchas actividades diarias tienden a confundirse con caídas y se vuelve frustrante la activación consecutiva de la alarma cuando en realidad no hay ningún peligro.

Mediante una máquina de soporte vectorial –SVM–, un acelerómetro y algunos mapas de profundidad, se plantea un sistema de detección de caídas de bajo costo. Usando estos dispositivos, los parámetros del suelo son calculados automáticamente, esto sumado a un algoritmo de clasificación de datos principalmente proveídos por el acelerómetro. El sistema, además, garantiza la privacidad de la persona al no ser intrusivo.

El conjunto de datos desarrollado es llamado UR Fall Detection Dataset –URFD–, y consta de imágenes de actividades del diario vivir, como caminar, sentarse, agacharse y acostarse. Se proveen, entonces, todas las muestras al clasificador para que este genere un autoaprendizaje y pueda determinar si una persona se encuentra o no en el suelo. Consta de 402 imágenes de

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 9 de 104

actividades típicas del diario vivir y 210 imágenes de representaciones de una persona en el suelo, además de 30 secuencias de imágenes con 30 caídas.

Gran parte de los sistemas de detección de caídas mediante dispositivos usables dependen de que el paciente en cuestión lo lleve puesto y que, además, esté consciente y en casa, por lo que no abarca el problema en su totalidad. Es en este punto cuando surge la pregunta de qué tanta garantía pueden dar los sistemas basados en dispositivos móviles. En 2015, Thomas Vilarinho *et al.* [16] analizan la precisión de un sistema de detección de caídas usando un *smartphone* y un *smartwatch* mediante el reconocimiento de patrones y umbrales. La investigación encontró que, después de suministrar datos de caídas simuladas y actividades del diario vivir, el sistema detectó de manera correcta el evento de caída en un 63 % y 78 % las actividades diarias.

El sistema es de código abierto, pensado para ser mejorado o inspirar nuevas investigaciones a futuro. Los sistemas desarrollados son:

1. El Umbral de aceleración del teléfono –PAT–
2. Reconocimiento del patrón del teléfono –PPR–
3. Reconocimiento de patrón y umbral del reloj –WTPR–

Sus aportes consisten en generar conocimiento para futuros sistemas, apuntando a reducir a largo plazo la tasa de muertes por caídas accidentales.

El trabajo a futuro es mejorar la representación de caídas, haciendo que las muestras que se tomen sean más reales y se acerquen más a verse como una caída verdadera. Al usar sujetos jóvenes simulando caídas, se genera una carencia de precisión en las caídas reales de ancianos; la velocidad con que cae un anciano difiere en gran medida a la velocidad en que cae una persona joven y sin ningún tipo de discapacidad o enfermedad.

Por otra parte, en 2017, Angela Sucerquia *et al.* [17] generan un dispositivo compuesto de dos tipos de acelerómetros y un giroscopio. Adicional, los autores generan un conjunto de datos

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 10 de 104

público que consta de 19 actividades del diario vivir (**ADV/ADL** [Activities of Daily Living]) y 15 tipos de caídas, realizado por 23 jóvenes adultos; 15 tipos de actividades del diario vivir realizadas por 14 adultos de más de 62 años con buen estado de salud, y datos de un participante de 60 años que realiza todas las ADL y caídas.

Más tarde, se logra crear un conjunto de datos más robusto disponible al público, realizando capturas de 2706 ADL y 1798 caídas. Se determinó que en estos dataset públicos no hay datos de personas que sufran alguna deficiencia física, es decir, todas las muestras que estos conjuntos de datos contienen son realizadas por personas saludables, esto significa que no tienen ningún tipo de discapacidad. Este dataset se da con el fin de usarse como *benchmark* por otros autores que estén realizando investigaciones en este mismo campo, con objeto de analizar los resultados usando sus propios algoritmos.

Kripesh Adhikari *et al.* [18], en 2017, presenta un sistema de detección de caídas basado en vídeo para los ambientes interiores como casas, apartamentos, etc. Lo más importante en cuanto a la detección de caídas es poder identificar correctamente las poses humanas y los diferentes cambios que hay entre ellas, ya que un cambio de posición específico del cuerpo puede definir o no una caída. Por ejemplo, la posición después de una caída es diferente a la posición de una persona al dormir o estar simplemente acostada; por ende, el sistema de detección de caída debe poder acertar sobre cuál postura está detectando. Los test se realizan en escenarios de interiores, como un cuarto de un hogar, el modelo propuesto no se ve afectado por la posición de la persona y puede enfrentarse a cualquier ángulo y forma de caminar alrededor del ambiente.

Usando redes neuronales convolucionales se logran identificar las diferentes posiciones. Se construye un dataset propio grabando diferentes actividades realizadas por diferentes personas en diferentes ambientes de interior. La combinación de un fondo con profundidad RGB junto a redes neuronales convolucionales es la mejor solución en el monitoreo de caídas mediante vídeo en los ambientes de interiores.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 11 de 104

Para esta investigación, únicamente se tuvo en cuenta los ambientes de interiores en hogares, siendo un sistema doméstico, descartando así los ambientes menos controlados, como hospitales u otros lugares encargados al cuidado de la salud de las personas. Por ello, el conjunto de datos realizado no podría ser entrenado en ambientes como los mencionados, puesto que no estaría en la capacidad de extraer los planos de profundidad con la misma precisión.

Nuevamente, en 2017, y de acuerdo con Moiz Ahmed *et al.*, [19] en países desarrollados han sido usados diferentes tipos de sensores para la detección de caídas en los adultos mayores, quienes son más vulnerable a estas, pero se necesitan mayores esfuerzos en países en vía de desarrollo donde no se implementan estos sensores o, incluso, no se reportan caídas. De esta forma, se propone un sistema de detección de caídas basado en clasificación, implementado mediante una red de sensores del área corporal. Como resultado, se desarrolla el dataset Smotion, el cual abarca varias posturas que pueden generar caídas en adultos mayores, usando una red del área corporal de sensores de luz y categorizando los datos en grupos de edad y peso.

Los métodos realizados en este estudio son tres. Primero, la creación del dataset, categorizando los ítems de este conjunto de datos en grupos de edad y peso; segundo, se desarrolla un sistema de clasificación y selección de características usando tres tipos diferentes de clasificadores máquina de soporte vectorial –SVM–, K vecinos más próximos –KNN– y una red neuronal –NN–; tercero, y último, la fabricación de un prototipo para generar alertas a diferentes tipos de servicios médicos o de emergencia en caso de alguna caída. A futuro se propone concentrarse en el uso de un kit de sensor de luz EMG para las estimaciones del riesgo de caída.

Finalizando el año 2017, Ramesh Rajagopalan *et al.* [20] realizan un análisis de algunos sistemas actuales de predicción y/o detección de caídas y determinan la carencia de tecnologías que puedan determinar el riesgo de caídas con base en los factores ambientales, es decir, solo se tienen en cuenta los ambientes controlados, discriminando así los no controlados. Para llevar a cabo este análisis, los autores evalúan los estados del arte de los diferentes sistemas de detección y predicción de caídas, además de tocar los puntos clave de estos sistemas, como, por ejemplo,

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 12 de 104

qué retos enfrentan en la sociedad, las limitaciones a tener en cuenta en los ambientes controlados en la mayoría de sistemas y cómo, a futuro, apuntar a mejorar el diseño e implementación de estos sistemas y aumentar la efectividad de los mismos.

Se logra determinar que, efectivamente, los sistemas existentes de predicción y detección de caídas son probados principalmente en ambientes controlados y dejan de lado los riesgos que los ambientes no controlados pueden suponer, dando un breve ejemplo: no se tienen en cuenta los múltiples factores dentro de un hospital que puedan generar una caída no intencional en un paciente. Las interacciones de las personas dentro de los ambientes no controlados son lo que principalmente generan eventos de caída: una baldosa en desnivel, un cable mal ubicado, piso mojado, entre otros.

Se propone el desarrollo de sistemas no intrusivos mediante dispositivos usables para el monitoreo de la presión sanguínea en pacientes, la fusión de datos de dispositivos usables y sensores del ambiente, diseño de interfaces de usuario, tener en cuenta los factores externos que generen riesgo de caída y comparaciones con evaluaciones clínicas en cuanto al riesgo de caídas.

En 2018, Manola Ricciuti *et al.* [21] proponen un método acertado para la detección de caídas mediante la investigación de marcos de profundidad del cuerpo humano usando una cámara de profundidad a cierta altura dentro de una habitación. Se lleva a cabo el desarrollo de un dataset que incluye 32 eventos de caídas y 8 actividades que se puedan interpretar como caídas. Un total de 800 secuencias realizadas por 20 adultos.

Las características que los autores obtienen mediante la extracción de los datos de los marcos de profundidad sirven para entrenar un clasificador basado en un algoritmo de aprendizaje con SVM dividido en tres fases:

1. Determinar una altura correcta a la cual debe ir la cámara de profundidad.
2. La extracción de características de los datos de profundidad.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 13 de 104

3. El procesamiento de los datos mediante el clasificador SVM.

Sin embargo, la precisión del sistema puede variar debido a que, en algunos casos, la persona que se encuentre en monitoreo puede no encontrarse en el rango de visión del dispositivo. Se propone, entonces, una posible futura solución que consta en implementar varios dispositivos Kinect, con el fin de cubrir más área, eliminar puntos ciegos y eliminar el ruido en los cuadros de profundidad. También se propone una opción a futuro para eliminar el ruido, aplicando técnicas de filtrado en pos-procesamiento.

Cabe resaltar que las muestras tomadas por los veinte adultos, si bien estos comprenden entre los 21 y 55 años, discriminan totalmente a la población que tenga alguna discapacidad o a los adultos mayores de 55 años, quienes son más vulnerables a sufrir lesiones graves por caídas, producto del deterioro de la edad.

Lourdes Martínez-Villaseñor *et al.* [22] realizan en el año 2019 un dataset público con el fin de abordar la falta de conjuntos de datos multimodales para el reconocimiento de actividades del diario vivir y la detección de caídas. Además, apuntan a colaborar con la motivación de investigaciones, con el fin de desarrollar más sistemas de detección de caídas, esta vez más robustos, que puedan reducir las consecuencias de las caídas accidentales en pacientes. Para esta investigación, se toman muestras de 17 individuos jóvenes sin ningún problema de salud donde realizan 11 actividades y caídas, con 3 intentos cada uno. El dataset resume más de 850 Gb de información de sensores portátiles, sensores ambientales y dispositivos de visión.

Para este estudio, hizo falta probar el conjunto de datos en un escenario no controlado, puesto que todas las muestras de eventos de caídas fueron capturadas usando protecciones para los autores y evitando alguna lesión, además de probar el sensor en personas diestras, discriminando así cualquier evento que pudiera darse en una persona zurda. Además, todas las pruebas fueron realizadas por pacientes jóvenes y sin algún tipo de discapacidad, lo cual aleja a las muestras de ser más semejantes a la realidad donde la gran mayoría de caídas accidentales ocurren en personas

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 14 de 104

de la tercera edad, quienes, en su mayoría, a causa de la edad, sufren de alguna discapacidad o deterioro del cuerpo, afectando sus actividades del diario vivir. A futuro, los autores proponen el desarrollo de un sistema de detección de caídas multimodal que pueda detectar un evento de caídas y emitir una alarma en tiempo real.

Muchos de los estudios para la creación o el planteamiento de sistemas para la detección de caídas solo toman en cuenta las posturas de pie, sentado y caído, pero no consideran las actividades diarias, actividades que pueden generar una caída en cualquier momento, especialmente en adultos mayores.

Finalmente, en 2019, Xiangbo Kong *et al.* [23] proponen un estudio de la efectividad de la cámara o sensor que sea usado dependiendo de la altura donde se instale, para poder determinar su precisión al momento de detectar una caída. El estudio propone un *tracking* del cuerpo y una red Alex-Net, con el fin de eliminar el ruido causado por el ambiente, y, de esta forma determinar de mejor manera cuando hay o no una caída.

Se crea un dataset público con aproximadamente 350 000 imágenes de profundidad en secuencias. Comprende 16 tipos de actividades diarias y 16 tipos de caídas realizadas por 8 personas en 8 direcciones y capturadas con una cámara de profundidad en 5 diferentes alturas. El sistema falla al momento de cambiar la altura de la cámara, puesto que cuando el sistema se entrena usando solo una altura específica previamente calculada con exactitud, su precisión es alta, pero hace que la robustez baje. Por otra parte, la presencia de mascotas o muebles dificulta el cálculo exacto de la altura a la cual debe ir esta cámara, puesto que obstaculiza el campo de visión.

A futuro, se propone el desarrollo de un algoritmo que permita calcular con exactitud la altura a la cual debe ir la cámara o sensor, pasando por parámetros todos los elementos a tener en cuenta en un ambiente de interiores, principalmente. Además, se plantea estudiar el impacto de los diferentes comportamientos de los individuos, así como crear un sistema que pueda detectar y

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 15 de 104

reconocer correctamente estos comportamientos para después determinar el mejor modelo a usar que pueda detectar caídas.

Estas investigaciones realizadas a lo largo de la última década, proveen elementos necesarios para creación de un nuevo y primer dataset robusto al brindar todas las características requeridas como la diversidad de escenarios y acciones, no limitado a muestras de solo una clase (ADL o FALL). Es con esta creación del conjunto de datos robusto que se busca hablar de un antes y un después, con respecto a la información disponible para los grupos de investigación interesados en el área de detección de caídas, esto tomando como soporte las carencias de los diversos datasets estudiados y resolviendo gran parte de estas en el conjunto de datos robusto consolidado al final del proyecto.

El resto del documento se encuentra distribuido de la siguiente manera partiendo del siguiente capítulo:

Capítulo 2: Marco Teórico.

Capítulo 3: Metodología CRISP-DM y sus respectivas fases.

Capítulo 4: Algoritmo de red neuronal.

Capítulo 5: Resultados de entrenamiento y validación.

Capítulo 6: Conclusiones.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 16 de 104

CAPÍTULO 2 - EXPLICACIÓN

2. MARCO TEÓRICO

2.1. Caídas

Según la OMS –Organización Mundial de la Salud– [1], una caída es la consecuencia de algún movimiento en el cual un paciente pierde el equilibrio y se precipita involuntariamente y de manera repentina hacia el suelo u otra superficie que pueda detenerlo. Estos eventos generan lesiones que pueden ser mortales. Algunas caídas pueden generar lesiones que terminan en algún tipo de discapacidad permanente.

Se calcula que anualmente se producen 646 000 caídas mortales en el mundo, convirtiéndolas en la segunda causa global de muerte por lesiones no intencionales (la primera son los accidentes de tránsito). En los países en vía de desarrollo es donde se registra más del 80 % de las muertes relacionadas con caídas a nivel mundial, donde un 60 % ocurren en regiones del Pacífico Occidental y Asia Sudoriental. Además, la mayor mortalidad por caídas es sufrida por adultos mayores de 60 años, quienes son más vulnerables a sufrir lesiones mortales.

Existen dos tipos de caídas. El primero, caídas en un mismo nivel, es el más frecuente y sus causas son principalmente resbalones o tropiezos. Estas caídas no suelen causar lesiones mortales, pero si discapacidades. El segundo tipo de caída es desde un nivel elevado, es decir, son causadas por caídas desde escaleras, edificios, vehículos, entre otros. Estas caídas ocurren con menor frecuencia, pero las lesiones en su mayoría son mortales.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 17 de 104

2.2. Dataset

En informática, un dataset o conjunto de datos es un grupo de archivos que contiene información que es procesada por aplicaciones especializadas en procesamiento de datos [24]. No cualquier aplicación tiene la facultad de realizar estos procesos, debido a la gran cantidad de datos contenidos.

Para este proyecto, los conjuntos de datos se conforman de clases, las cuales son los estados que se desea monitorear; en este caso, son dos clases específicas: caída y no caída. Los registros o muestras, que son los archivos que serán usados en el aprendizaje profundo por parte del algoritmo que se implemente en el sistema, pueden ser de valores numéricos en los cuales se registran coordenadas, datos del ambiente y del paciente o también secuencias de imágenes que permitirán detectar los cambios en la posición de una persona mediante el reconocimiento de la pose del cuerpo humano por medio de una cámara o sensor, así como características que van desde el tamaño, los tipos de archivos que conforman el dataset, entre otros.

Hay diferentes tipos de dataset: los convencionales son aquellos conjuntos de datos cuya información es proveída por dispositivos usables, donde se registran coordenadas del cuerpo humano en movimiento para, de esa forma, determinar una caída, así como los datos de la persona que use el dispositivo, como el pulso cardíaco y demás datos que necesite el sistema en cuestión que esté en uso. Por otra parte, el segundo tipo de dataset es aquel de imágenes o de vídeos, los cuales usan sensores o cámaras para realizar una captura en tiempo real del cuerpo humano, basándose en las acciones o movimientos que realice [24].

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 18 de 104

Los dataset estructurados, o data estructurada, se pueden pensar como una serie de información organizada por columnas, como si se tratase de una base de datos, es decir, donde la data se encuentra organizada por diferentes campos.

train_id		name	item_condition_id	category_name	brand_name	price	shipping	item_description
0	0	MLB Cincinnati Reds T Shirt Size XL	3	Men/Tops/T-shirts	NaN	10.0	1	No description yet
1	1	Razer BlackWidow Chroma Keyboard	3	Electronics/Computers & Tablets/Components & P...	Razer	52.0	0	This keyboard is in great condition and works ...
2	2	AVA-VIV Blouse	1	Women/Tops & Blouses/Blouse	Target	10.0	1	Adorable top with a hint of lace and a key hol...
3	3	Leather Horse Statues	1	Home/Home Décor/Home Décor Accents	NaN	35.0	1	New with tags. Leather horses. Retail for [rm]...
4	4	24K GOLD plated rose	1	Women/Jewelry/Necklaces	NaN	44.0	0	Complete with certificate of authenticity

Figura 1. Ejemplo de data estructurada.
Tomada de: Towards Data Science.

En la Figura 1 se ve un ejemplo de data estructurada, en este caso, para una tienda *online*. Este está organizado por columnas, y, si se hiciese una búsqueda de algún dato, fácilmente se encontraría el dato deseado. [25]

Por otro lado, la data no estructurada es toda aquella que no tiene un modelo o estructura predefinida. Un claro ejemplo de este tipo de data son las imágenes, ya que, al ser un conjunto de píxeles, no es posible organizar la información que estos contienen en una tabla como la anterior. [26]

Son muchos los datasets usados para determinar si un algoritmo de red neuronal se desempeña correctamente en cuanto a la detección y el reconocimiento de objetos. Para el presente proyecto, son necesarios los algoritmos que ejecutan estas detecciones y reconocimientos, puesto que el dataset realizado es de clasificación. Para esto se debe detectar, dependiendo de una secuencia de imágenes, si la acción en dicha secuencia es una actividad del diario vivir –ADL– o una caída –Fall–.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 19 de 104

Un par de ejemplos de dataset más usados en *benchmark* son:

2.2.1. Common Objects in Context (COCO)

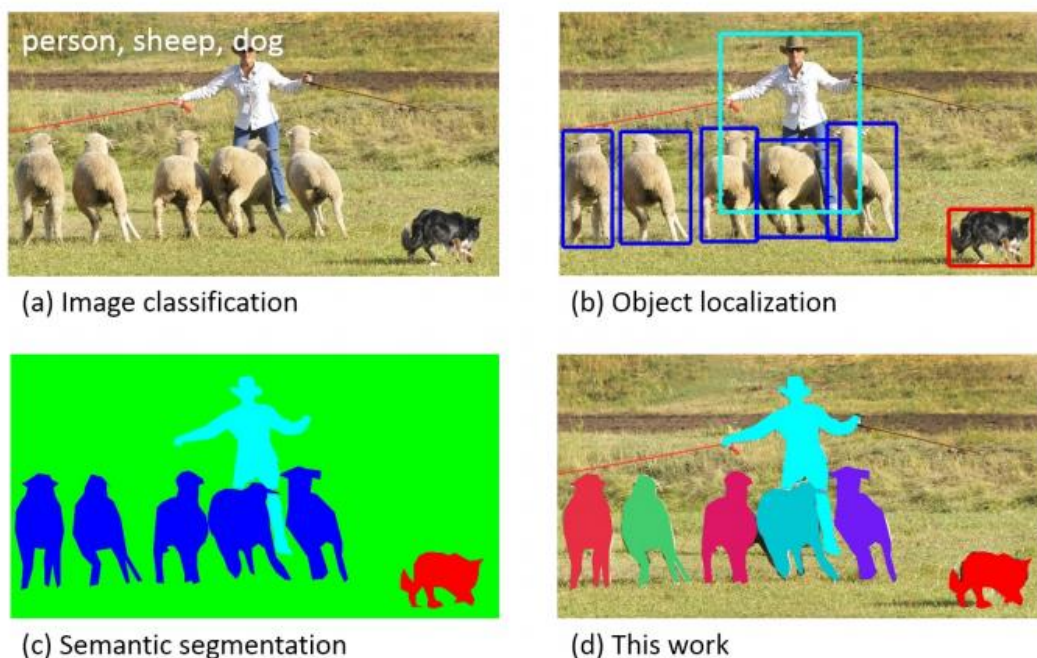


Figura 2. Detección de objetos
Tomada de: Microsoft COCO: Common Objects in Context

Tsung-Yi Lin *et al.* [27] presentan un dataset en 2014 (Figura 2), con el fin de lograr identificar objetos que se encuentran en el contexto natural, es decir, en el diario vivir, objetos dentro de escenas en las que comúnmente una persona se ve implicada. Las imágenes que componen el dataset contienen 91 tipos de objetos que pueden ser identificados fácilmente por un niño de cuatro años, esto significa 2,5 millones de instancias, etiquetadas en 328 000 imágenes en total.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 20 de 104

2.2.2. ImageNet

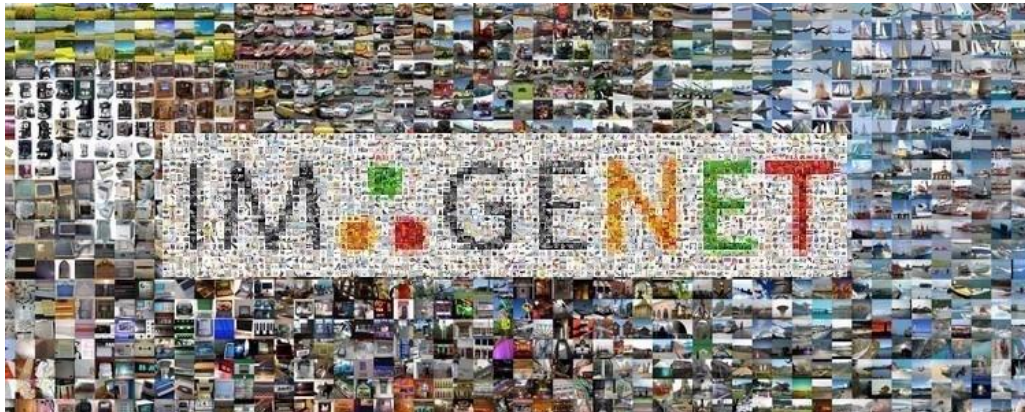


Figura 3. ImageNet Dataset
Tomada de: cv.glucon.ai

En 2009, Jia Deng *et al.* [28] crean un dataset de imágenes de gran tamaño, compuesto de más de 14 millones de imágenes (Figura 3) comprendidas entre objetos, plantas, entre otros, clasificadas en más de 20 000 categorías. Este conjunto de datos, al ser tan robusto, es un excelente *benchmark* para los algoritmos de clasificación, debido a que demuestra qué tan bien aprende o no la red neuronal a identificar diferentes tipos de objetos, paisajes, plantas, etc.

En el presente proyecto, los dataset usados para la construcción del Robust Fall Detection Dataset –RFDD– fueron: BOSS Dataset [29], Cogent Labs [30], Fall Detection Dataset [31], SisFall Dataset [32], Evaluating Depth Dataset [33], UR Fall Detection Dataset [15] y UMA Fall Dataset [34].

2.3. Deep learning

El *deep learning*, o aprendizaje profundo, es un tipo de técnica de inteligencia artificial en el cual los algoritmos tienen la capacidad de auto entrenarse para realizar tareas sin necesidad de una programación específica, es decir, que sea capaz de imitar lo que un ser vivo inteligente haría en la vida real al procesar información y poder identificar situaciones, diferencias entre objetos, entre otros. Para esto, se usa una red neuronal artificial compuesta de diferentes niveles; entre más

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 21 de 104

niveles, más profundo es el aprendizaje. Si bien puede brindar un mejor resultado en algunos casos, el tiempo se verá incrementado considerablemente, debido a que el algoritmo deberá realizar más procesos [35].

2.4. Redes neuronales convolucionales

Son un tipo de redes neuronales que están adaptadas principalmente a la visión por computadora, gracias a su habilidad para abstraer de manera jerárquica representaciones por medio de diferentes medios, como el vídeo. La arquitectura de estas redes tiene como base el neocognitron, introducido por Kunihiko Fukushima en 1980, el cual toma como referencia el modelo procesamiento que se lleva a cabo en la corteza visual propuesto por Hubel y Wiesel.

La red consiste en una capa de entrada compuesta por una matriz de fotorreceptores, seguida de una amplia cantidad de conexiones de estructuras modulares, cada una compuesta por dos capas de celdas conectadas una debajo de otra. Esta red tiene la habilidad de auto aprender o auto entrenarse sin necesidad de una supervisión [36]–[38].

2.5. Capa convolucional

Esta capa es la primera en usarse para la extracción de características o muestras de las imágenes de entrada del modelo de red. La convolución matemática se ejecuta entre la imagen de entrada y un filtro de tamaño $M \times M$. Al aplicar el filtro sobre la imagen de entrada, el resultante es la aplicación del filtro y a las partes de la imagen de entrada con respecto al tamaño del filtro ($M \times M$). La salida de esta capa es un conjunto de características que proporcionan la información respecto a la imagen. Este conjunto de características es alimentado por más capas del modelo para, de esta forma, realizar un aprendizaje sobre las características de la imagen que se haya pasado por entrada. [39]

2.6. Capa pooling

En la mayoría de los casos, esta capa sucede a la convolucional, y su trabajo es reducir el tamaño del mapa de características para que, computacionalmente, se reduzca la carga de complejidad de

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 22 de 104

procesado. Esto se realiza disminuyendo las conexiones entre otras capas y se ejecuta de forma independiente en cada mapa de características. [39]

2.7. Capa fully-connected (FC)

Se compone de los pesos y sesgos entre neuronas, y sirve para conectar neuronas con dos capas diferentes. Normalmente, están antes de la salida de la arquitectura de red neuronal. Es aquí cuando la imagen que ha venido procesándose desde otras capas es aplanada y alimenta las capas FC, donde se realizan operaciones matemáticas y se inicia el proceso de clasificación. [39]

2.8. Capa dropout

Una vez todas las características están sujetas a la capa FC, se puede dar un sobreajuste en los datos de entrenamiento. Este sobreajuste se suele provocar por un muy buen rendimiento del modelo de red, pero esto provoca, al mismo tiempo, un efecto negativo, ya que el modelo se vuelve incapaz de aprender nuevas características y tomará los nuevos datos como los anteriores. Esta capa elimina algunas neuronas que han estado en el proceso de entrenamiento, reduciendo el tamaño del modelo. [39]

2.9. Funciones de activación

Una función de activación sirve para determinar qué información debe continuar el flujo del modelo de red hacia su salida, es decir, qué información debe seguir avanzando y qué información no. Hay varios tipos de activación: ReLU, Softmax, tanH y Sigmoid. [39]

2.10. Arquitecturas de red neuronal usadas en el proyecto

2.10.1. VGG16

Propuesto por Karen Simonyan y Andrew Zisserman en 2014, es un modelo de red neuronal convolucional pre-entrenado.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 23 de 104

El modelo logró un 92,7 % de precisión de las 5 primeras pruebas en ImageNet, siendo presentado en la competición ILSVRC-2014. VGG16. Está constituido por 16 capas + 1 capa extra de salida SoftMax con diferentes pesos:

1. Capa de convolución usando 64 filtros.
2. Capa de convolución usando 64 filtros + *max pooling*.
3. Capa de convolución usando 128 filtros.
4. Capa de convolución usando 128 filtros + *max pooling*.
5. Capa de convolución usando 256 filtros.
6. Capa de convolución usando 256 filtros.
7. Capa de convolución usando 256 filtros + *max pooling*.
8. Capa de convolución usando 512 filtros.
9. Capa de convolución usando 512 filtros.
10. Capa de convolución usando 512 filtros + *max pooling*.
11. Capa de convolución usando 512 filtros.
12. Capa de convolución usando 512 filtros.
13. Capa de convolución usando 512 filtros + *max pooling*.
14. Capa *fully-connected* con 4096 nodos.
15. Capa *fully-connected* con 4096 nodos.
16. Capa *fully-connected* con 1000 nodos.
17. Capa de salida Softmax con 1000 nodos. [40]

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 24 de 104

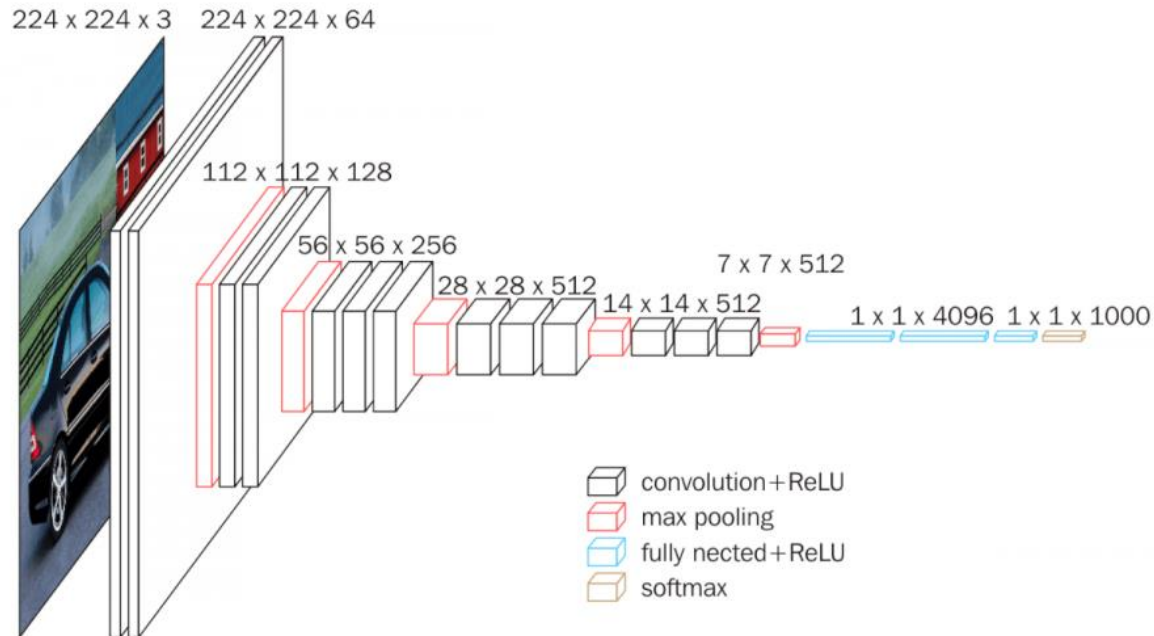


Figura 4. Modelo de red neuronal VGG16
Tomada de: neurohive.io.

La capa de entrada recibe como parámetro una imagen de tamaño 224x224px (Figura 4). Posteriormente, la imagen pasa a través de dos capas convolucionales de tamaños 224x 224 x 64px, seguido de un *max pooling*, cuyo trabajo es redimensionar la imagen a 112x112x64px. Las siguientes dos capas constan de tamaño 112x112x128, y nuevamente un *max pooling* que redimensiona la imagen a 56 x56x128. El proceso se repite ahora con tres capas de 56x56x256 y *max pooling* para redimensionar a 28x28x256. La imagen pasa por otras tres capas convolucionales de 28x28x512 hasta el *max pooling* que ahora redimensiona a 14x14x512. El proceso es el mismo en las siguientes tres capas, y luego la imagen es redimensionada a 7x7x512. En este punto, la imagen pasa por dos capas densas o *fully-connected*, cada una de 4096 nodos, y una siguiente también *fully-connected*, con 1000 nodos. Cada nodo sirve para una clase diferente, en este caso, esta arquitectura, al haber sido presentada para resolver el dataset ImageNet, que consta de 1000 clases, actúa para ajustarse al mismo. La última capa es una capa SoftMax. Esta

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 25 de 104

configuración de las capas *fully-connected* actúa de la misma manera en cualquier red neuronal. [41], [42]

2.10.2. ResNet

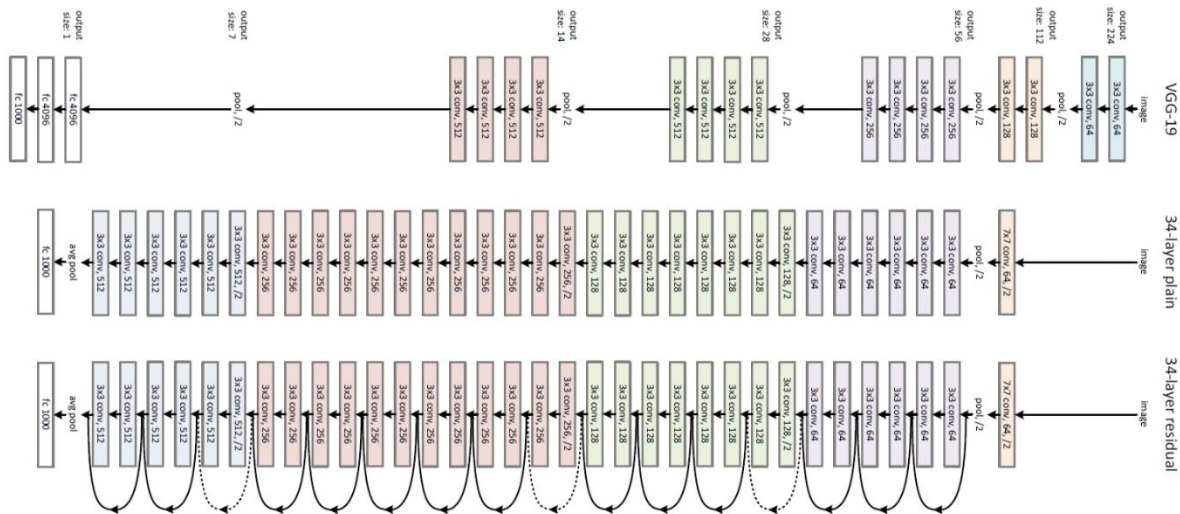


Figura 5. Modelo de red neuronal ResNet
Tomada de: medium.com

ResNet o Residual Networks –Redes Residuales– es una arquitectura de red neuronal propuesta por Kaiming He *et. al* [43], presentada en 2015 en las competencias ILSVRC 2015 y COCO 2015, resolviendo el dataset ImageNet con una tasa de error del 3,57 %, y reemplazando las capas de VGG16 en velocidad con ResNet101, observándose una mejora del 28 %. Este modelo usa una red de 34 capas inspirada en VGG19 (Figura 5), en donde luego se agrega un acceso directo de conexión, lo que omite el entrenamiento de una o más capas, generando así un bloque residual. La arquitectura propone un bloque residual optimizado, añadiendo una extensión conocida, como cuello de botella, que puede reducir la dimensión en las primeras capas convolucionales y luego volver a incrementarse al final de la capa convolucional. [44], [45]

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 26 de 104

Para este proyecto, se hace una validación con ResNet50 y ResNet101. La diferencia entre estas dos consiste en el número de capas que usa cada una, donde ResNet50 usa 50 capas, y ResNet101, 101 capas.

2.10.3. InceptionV3

Es la tercera versión de la serie de arquitecturas de redes neuronales profundas desarrolladas por Google llamadas Inception. Esta versión es presentada en 2015 y pre-entrenada usando el dataset compuesto por 1000 clases ImageNet [46]. Consta de 13 capas (Figura 6):

1. Capa convolucional (299x299x3).
2. Capa convolucional (149x149x32).
3. Capa convolucional con *padding* (147x147x32).
4. *Pool* (147x147x64).
5. Capa convolucional (73x73x64).
6. Capa convolucional (71x71x80).
7. Capa convolucional (35x35x192).
8. Capa 3xInception (35x35x288).
9. Capa 5xInception (17x17x768).
10. Capa 2xInception (8x8x1280).
11. *Pool* (8x8x2048).
12. Capa linear (1x1x2048).
13. Capa SoftMax (1x1x1000). [47]

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 27 de 104

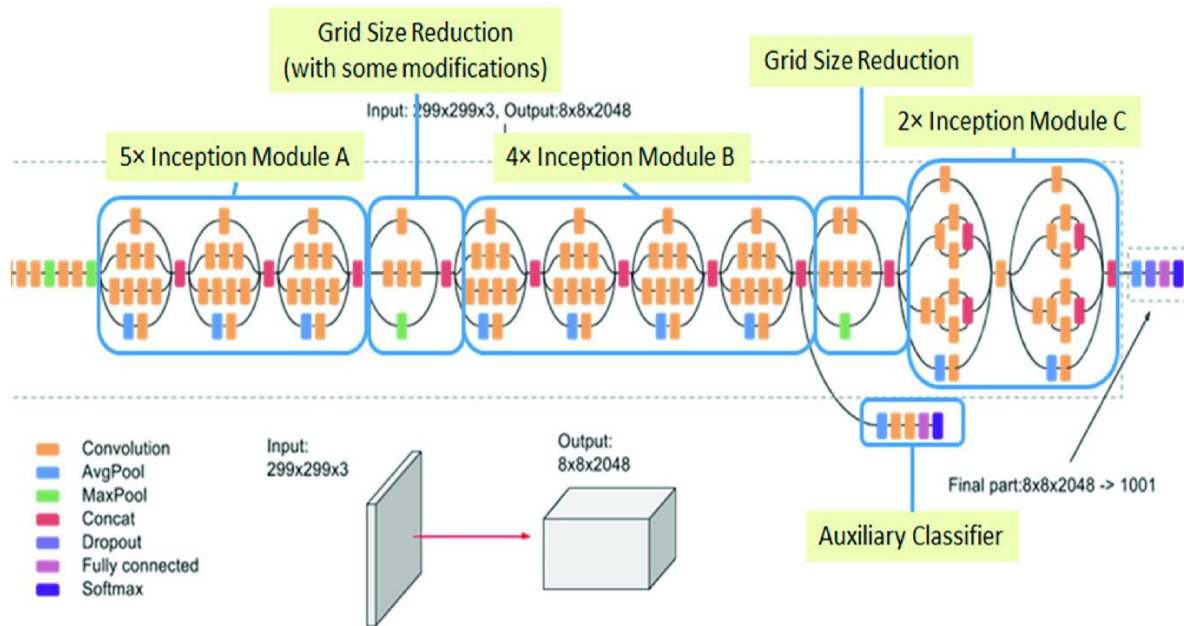


Figura 6. Modelo de red neuronal Inception
Tomada de: iust-projects.ir.

Inception busca reducir el cuello de botella. En esta versión se trabaja con la premisa de que las redes neuronales se desempeñan mejor cuando las convoluciones no hacen redimensiones a las imágenes de una manera muy drástica, ya que, al reducir demasiado las dimensiones, puede generar una pérdida de información, y a esta consecuencia se la conoce como cuello de botella, al disponer de poca información y mucho proceso para aplicarle a la data. Al usar métodos de factorización, las convoluciones son más eficientes en cuanto a procesamiento computacional.

La solución planteada es factorizar la convolución 5x5 a dos operaciones convolucionales de 3x3, con el fin de mejorar la velocidad de procesado. Lo anterior se factoriza nuevamente en convoluciones de tamaño $n \times n$ a una combinación de convolución $1 \times n$ y $n \times 1$. Al requerir una convolución de 3x3, primero se hará una operación convolucional de 1×3 , y después de 3×1 . Los desarrolladores de Inception encontraron este método 33 % más eficaz que hacer una sola convolución de 3x3. La versión 3 de Inception, que fue la usada en la validación para este proyecto,

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 28 de 104

añade, además, la factorización en convoluciones 7x7, agilizando el procesamiento de información. [48]

2.10.4. Xception

Arquitectura de red neuronal profunda desarrollada por Google y presentada en 2016 como una interpretación de los módulos de Inception, y como una alternativa para poder separar las operaciones convolucionales en profundidad. Esto se puede entender como un módulo de Inception con un número máximo de torres. Como se explica, Xception es inspirada en Inception, donde los módulos de esta última son reemplazados por convoluciones en profundidad separadas. Para entenderlo mejor, puede decirse que se separa en bloques convolucionales. [49]

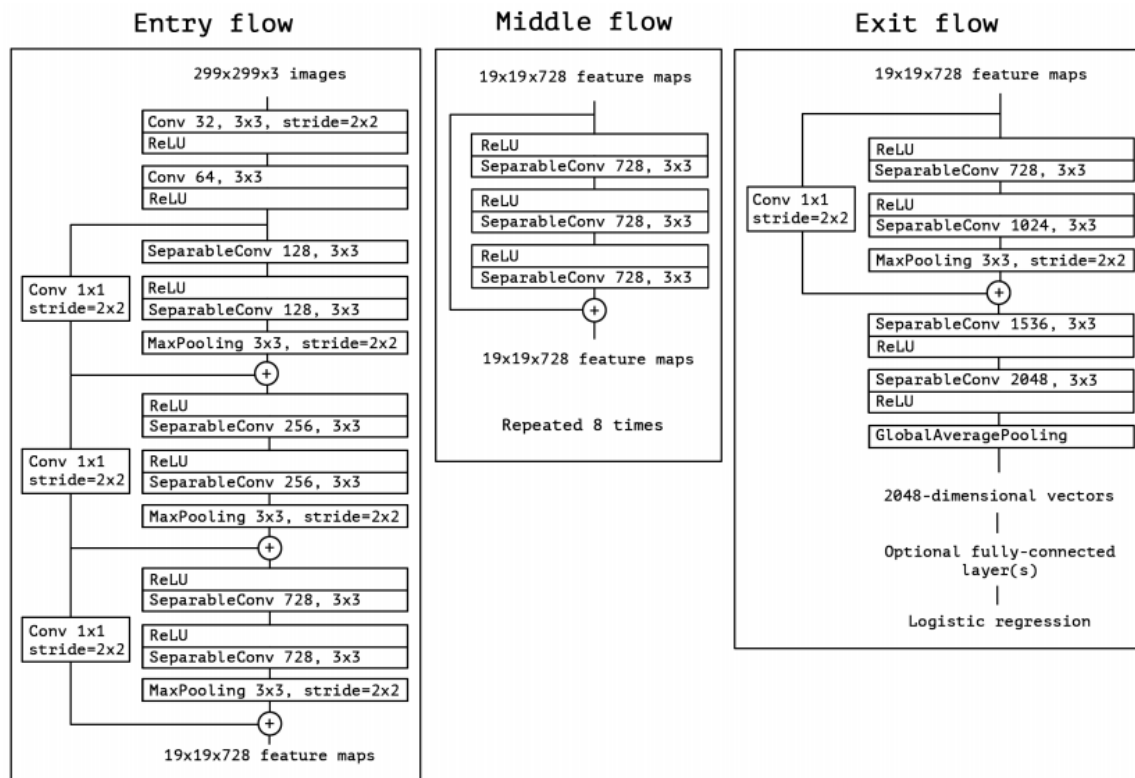


Figura 7. Capas del modelo de red neuronal Xception
Tomada de: Towards Data Science.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 29 de 104

Xception se compone por tres partes: el flujo de entrada, el flujo medio (que se repite ocho veces) y el flujo de salida (Figura 7):

El flujo de entrada consta de dos capas convolucionales ($3 \times 3 \times 32$ y $3 \times 3 \times 64$, respectivamente) seguidas de una capa convolucional separable¹ ($3 \times 3 \times 128$) y una capa convolucional separable ReLU, también convolucional separable ($3 \times 3 \times 128$), además de un *MaxPooling* después de cada bloque. Se repiten las capas convolucionales separables ReLU dos veces más con tamaños de $3 \times 3 \times 256$ y $3 \times 3 \times 728$, correspondientemente con sus *MaxPooling* al final. Del mismo modo que ResNet, Xception maneja conectores de salto que omiten algunos entrenamientos.

El flujo medio se compone de tres capas convolucionales de separación ($3 \times 3 \times 728$), que se repiten ocho veces y tienen la opción de añadir una capa *fully-connected* al final.

El flujo de salida posee dos capas convolucionales de separación ReLU ($3 \times 3 \times 728$ y $3 \times 3 \times 1024$ respectivamente) con *MaxPooling* al final; luego, dos capas convolucionales de separación ReLU ($3 \times 3 \times 1536$ y $3 \times 3 \times 2048$, en ese orden) y un *Pooling* de promedio global, 2048 vectores dimensionales, capas *fully-connected* opcionales y una regresión logística al final. Este flujo también cuenta con un conector de salto. [50]

2.10.5. DenseNet

En 2017, Gao Huang *et al.* [51] introducen DenseNet, que conecta cada capa con todas las demás capas creando una retroalimentación, es decir, cambia el esquema de tener una cantidad igual de capas y conexiones. Para esta arquitectura, por cada capa, todas las conexiones de capas

¹ Las convoluciones separables consisten en realizar primero una convolución espacial en profundidad, que actúa en cada canal de entrada por separado, seguida de una convolución, puntual que mezcla los canales de salida resultantes. [65]

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 30 de 104

predecesoras son usadas como entradas, y todo este conjunto es usado como entradas en las capas posteriores.

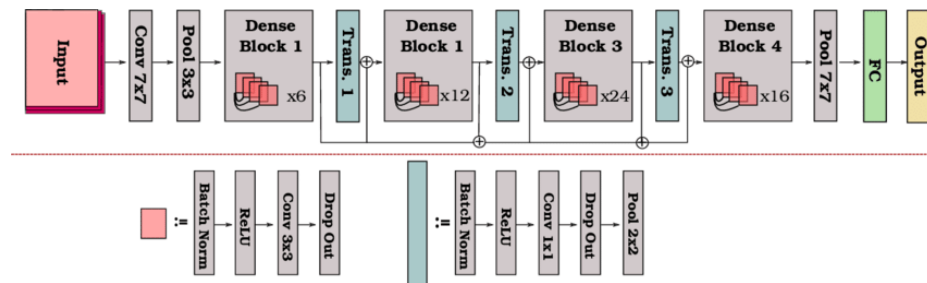


Figura 8. Modelo de red neuronal DenseNet
Tomada de: ResearchGate.

Esta arquitectura se compone de (Figura 8):

- Capas de composición, donde actúan la preactivación, la normalización por lotes –BN– y ReLU, seguido de una convolución 3x3.
- Capas de cuello de botella, que reducen la complejidad y tamaño del modelo. Se usa la normalización por lotes y ReLU, seguidos por una convolución 1x1, y después BN, ReLU y convolución 3x3.
- Múltiples bloques de densidad con capas de transición: convoluciones de 1x1, seguidas por agrupación de promedios (*average pooling*), cuyo papel es el de capas de transición entre dos bloques de densidad juntos. [52]

2.10.6. *EfficientNetB7*

Mingxing Tan *et al.*, investigadores de Google, publican en 2019 una nueva familia de redes neuronales llamadas EfficientNet que, como su nombre lo sugiere, son más eficientes a nivel de procesamiento computacional. Con EfficientNetB7, que sería la séptima entrega de la familia de redes Efficient, se logra una precisión del 84,3 % en el top 1, y 97 % en el top 5 (en este proyecto se testea el dataset RFDD con EfficientNetB7, es por ello que los resultados mostrados son de B7) al resolver el dataset ImageNet [53].

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 31 de 104

Esta arquitectura usa un método que hace un escalado a las imágenes de manera uniforme a todas las dimensiones de profundidad, ancho y resolución, usando un coeficiente compuesto. Este coeficiente se sugiere debido a la premisa de imágenes de entrada muy grandes, esto supone a la red neuronal el uso de más capas y, por ello, más complejidad computacional.

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Figura 9. Capas del modelo de red neuronal Efficient Net
Tomada de: Learn OpenCV.

EfficientNet se compone de nueve bloques en total:

1. Bloque con capa convolucional 3x3 [224x224].
2. Bloque MB* [112x112].
3. Bloque MB [112x112].
4. Bloque MB [56x56].
5. Bloque MB [28x28].
6. Bloque MB [14x14].
7. Bloque MB [14x14].
8. Bloque MB [7x7].
9. Bloque con capa convolucional 1x1 y capas de *Pooling* y *Fully-Connected* [7x7]. [54], [55]

* Los bloques residuales invertidos o MB son tipos de bloques residuales usados por los modelos de imágenes que usan estructura inversa para mejorar la eficiencia. Estos bloques provienen de la arquitectura MobileNetV2. [56]

2.10.7. MobileNetV2

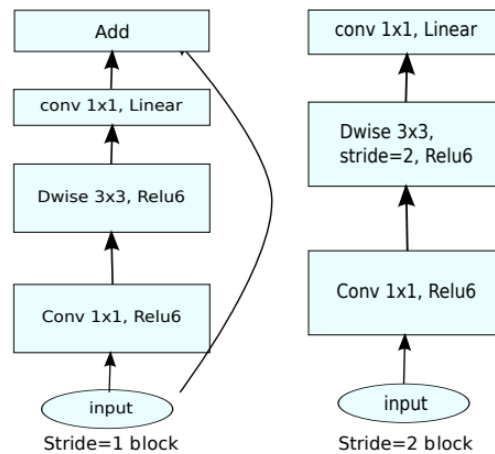


Figura 10. Modelo de red neuronal MobileNetV2
Tomada de: Opengeni.us.org.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	-

Figura 11. Capas del modelo de red neuronal MobileNetV2
Tomada de: Towards Data Science

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 33 de 104

MobileNetV2 es una arquitectura de red neuronal convolucional pensada para tener un buen rendimiento en dispositivos móviles. Su base es una estructura residual, en la cual las conexiones residuales se encuentran entre las capas de cuello de botella (figuras 10 y 11). Mark Sandler *et al.* [56] publican en 2019 esta versión de MobileNet, mencionándola como una nueva arquitectura para dispositivos móviles.

Existen dos tipos de bloques: un bloque residual con *stride* 1, y otro bloque con *stride* 2 para el redimensionamiento. Hay tres capas para cada bloque, la primera capa es convolucional ReLU6, la segunda es de convolución profunda y la tercera es convolucional sin ninguna no linealidad. [56]

2.11. Métricas de clasificación

Para el presente proyecto, se abordaron las siguientes métricas de clasificación: exactitud, precisión, matriz de confusión, función de pérdida y coeficiente Kappa. La exactitud se puede medir en la cantidad de resultados positivos en una cantidad de casos analizados. La precisión, por su lado, es la proporción de esa exactitud, que realmente es positiva, y se usa para tener seguridad sobre una predicción hecha. [57]

Las métricas de clasificación en general permiten conocer qué tan certero es un algoritmo de red neuronal cuando se usa para resolver algún conjunto de datos [58]. Por ejemplo, el reto que todos los algoritmos han tenido a lo largo de los años: resolver el dataset ImageNet. Si bien muchas arquitecturas han conseguido buenos resultados en cuanto a precisión y exactitud, aún no hay ninguna que logre un 100 % en ambas métricas mencionadas. Estas clasificaciones se realizan para diferenciar entre dos o más clases, dependiendo del conjunto de datos que se maneje. Para entender mejor el concepto de las clases, se puede dar un ejemplo con animales: una clase puede ser los mamíferos; otra clase, reptiles; otra clase, insectos; entre otros. Las métricas de clasificación nos sirven, en este caso, para determinar la precisión y exactitud de identificación de un animal en alguna imagen, poder clasificarlo en alguna de esas clases (una buena clasificación sería poner una

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 34 de 104

imagen de un oso en la clase de mamíferos). Se dice que las métricas están con un porcentaje alto en cuanto a exactitud y precisión, caso contrario si un caimán es clasificado como un insecto.

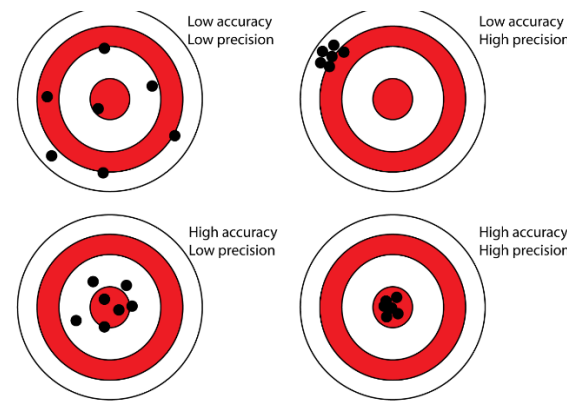


Figura 12. Diagrama de comparación exactitud y precisión

Tomado de: antarcticglaciers.org.

Para ilustrar mejor la diferencia entre precisión y exactitud, se encuentra el ejemplo de la diana (Figura 12). Se entiende por precisión cuando un resultado tiende a ser siempre el mismo, independiente de si está errado o no. La exactitud, por su parte, es cuánto se acerca un resultado a la realidad: de una lista de 10 de estos, por ejemplo, si se busca un resultado de 15 y se tiene 13, 14, 16, 17, 15, 14, 14, 17, 17, 15, entonces podrá afirmarse que tiene alta exactitud, ya que se aproxima en gran medida al resultado deseado y no tiene alta precisión, debido a que varían los valores con respecto al esperado.

Una matriz de confusión se trata de una tabla con cuatro combinaciones diferentes (Figura 13), donde se encuentran los valores reales, es decir, la data real, y los datos predichos, es decir, aquellos que arroja nuestro modelo de red después de haber hecho su debido aprendizaje con la información que se haya ingresado.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 35 de 104

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figura 13. Matriz de confusión, True Positive - TP (Verdadero Positivo): Predicción positiva y valor real positivo. True Negative - TN (Verdadero Negativo): Predicción negativa y valor real negativo. False Positive - FP (Falso Positivo): Predicción positiva y valor real negativo. False Negative - FN (Falso Negativo): Predicción negativa y valor real positivo. Fuente: bookdown.

La función de pérdida es la distancia medida entre la salida que arroja el modelo y la salida esperada por el investigador, esto determina cómo el algoritmo usado modela los datos. Puede ser categorizada en dos: clasificación para valores discretos y para regresión con valores continuos. [59]

En algunos casos, el investigador se encuentra frente a problemas de clasificación entre varias clases. Muchas veces no basta con la precisión y exactitud para poder definir el desempeño de un modelo de red, ya que algunas capas pueden fallar en la predicción. El coeficiente Kappa es una medida de concordancia para elementos cualitativos que ajusta el efecto de azar en la concordancia resultante. [60]

El dataset de este proyecto está comprendido entre métricas de clasificación, debido a que se tienen dos clases: actividades del diario vivir –ADL– y caídas –Fall–. El dataset reúne gran cantidad de información sobre estas clases en vídeos que posteriormente serán procesados por un algoritmo de red neuronal que extraerá diferentes muestras para entrenamiento, de este modo, luego se podrá determinar si hay una actividad del diario vivir o una acción de caída mediante una entrada de vídeo o imagen, dependiendo de la necesidad del investigador.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 36 de 104

CAPÍTULO 3 – DESARROLLO DEL PROYECTO

Para la construcción del conjunto de datos, se opta por utilizar Crisp-DM (Ver Figura 14), por su calidad final que ofrece al ser un proceso que se encarga de la minería de datos de manera independiente al desarrollo del algoritmo para procesar los datos [61], [62]. Para el conjunto de datos, se requirió estudiar gran cantidad de muestras y hacer una depuración de la información, por lo que, para entender con mayor facilidad y eficiencia el proceso de recolección de datos, esta metodología crea varias etapas mostradas en la Figura 14:

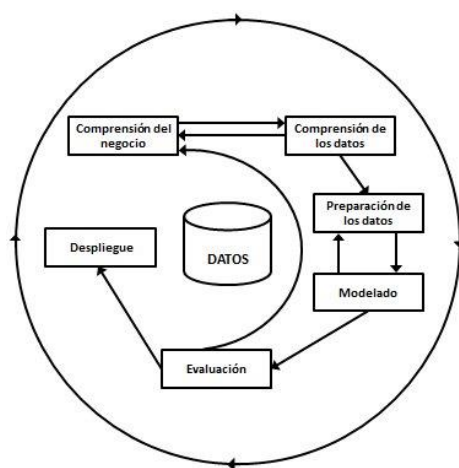


Figura 14. Metodología CRISP-DM [61], [62]

A continuación, se presentan los pasos descritos en la Figura 14, para la aplicación de la metodología en el proyecto de investigación.

3. METODOLOGÍA

3.1. Fase I

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 37 de 104

Uno de los mayores problemas para los investigadores es el tema del dataset al ser usado para las pruebas de sus algoritmos. La creación de un conjunto de datos desde cero supone una gran inversión en tiempo y dinero, ya que se debe tener actores que realicen las acciones necesarias, sean simulaciones de caídas o de actividades del diario vivir –ADL–. Algunos investigadores podrán tener los actores como parte de su equipo de trabajo sin tener que gastar dinero en actores externos, pero no disponer del tiempo suficiente para realizar las tomas de las acciones necesarias. Puede ocurrir un escenario inverso o simplemente el equipo de investigación no dispone de tiempo para la creación de un nuevo dataset ni de los recursos para contratar actores que realicen las acciones que componen el mismo.

ARQUITECTURAS DE RED RESOLVIENDO IMAGENET

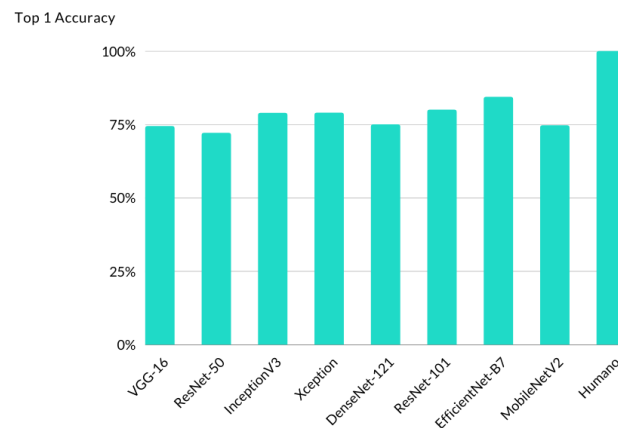


Figura 15. Arquitecturas de redes neuronales convolucionales aplicadas en la resolución del conjunto de datos ImageNet. Fuente: paperswithcode [63].

Muchos dataset robustos son usados como *benchmark* para las distintas arquitecturas de red que existen, al igual que con los diferentes algoritmos creados por grupos de investigación en la detección de caídas. A lo largo de los años, han surgido nuevas arquitecturas de red capaces de resolver estos dataset robustos con el objetivo de poder igualar cada vez más la precisión y exactitud que un humano puede arrojar en cuanto a la clasificación de objetos (ver Figura 15).

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 38 de 104

La creación de nuevas arquitecturas de redes neuronales día tras día perfecciona más estas métricas, asemejándose cada vez más a un ser humano. Se puede ver una comparación de AlexNet (Figura 16), una de las arquitecturas de red neuronal más importantes a principios de la segunda década del 2000, siendo superada por algunas arquitecturas creadas posteriormente.

Model	Top-1 accuracy	Top-5 accuracy
AlexNet	0.625	0.86
VGG-16	0.715	0.901
Inception	0.782	0.941
ResNet-152	0.870	0.963
Xception	0.790	0.945

Figura 16. Precisión de clasificación de AlexNet, VGG-16, ResNet-152, Inception y Xception en ImageNet

La importancia de un dataset lo suficientemente robusto radica en el desafío del investigador por poder resolver dicho conjunto de datos, es decir, estar en condiciones de crear un algoritmo capaz de identificar las diferentes clases del conjunto de datos, y no al contrario, que sea el investigador quien tenga un algoritmo bastante eficiente, pero no disponga de un conjunto de datos que pueda igualarse a la calidad del algoritmo.

Es por esto que, al igual que ImageNet, la idea de crear un dataset lo suficientemente robusto, pero en el campo de la detección de caídas, es bastante importante para que las investigaciones puedan acelerar su proceso por los motivos explicados en el primer párrafo de esta fase de CRISP-DM.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 39 de 104

3.2. Fase II

Cuando un grupo de investigación que construye un dataset propio, se puede decidir si ponerlo en circulación de manera comercial o de manera gratuita, claramente, con reconocimiento al autor (*creative commons*). Como el proyecto busca reducir costos, tanto de tiempo como económicos, se opta por reunir dataset públicos.

La gran mayoría de dataset usados para la construcción del RFDD (**Robust Fall Detection Dataset**), a pesar de estar bien en cuanto a las acciones realizadas por los actores, tenían falencias en cuanto a las secuencias de imágenes. Al respecto, se encontraron muchos “fragmentos muertos” (ver Figura 18), llamados así por tener al actor en una misma posición después de haber realizado alguna acción continuando con la grabación del vídeo, generando un archivo de mayor duración y tamaño.

Además, de los fragmentos muertos, se hallaron muchos fragmentos vacíos (ver Figura 17), donde el vídeo continuaba reproduciéndose, pero no había algún tipo de acción realizada por actores, resultando inútil a la hora de realizar un entrenamiento con redes neuronales, puesto que esta información extraída de un vídeo entorpecería el proceso de aprendizaje de la red.

Por último, dos o más acciones realizadas por el o los actores, se podían encontrar en un mismo vídeo (ver Figura 19). El problema de realizar un entrenamiento con estas acciones múltiples existe cuando la investigación requiera solo un tipo de acción específica, como caminar o sentarse, al evidenciarse más de una acción en un vídeo, y estas siendo diferentes unas de otras (caminar, sentarse, levantarse, etc.), en este caso, el entrenamiento no podría lograrse como es requerido y supondría al equipo de investigación una pérdida de tiempo al tener que recortar mediante edición de vídeo solo la acción necesaria o creando un algoritmo que solo extraiga muestras para entrenamiento en determinado fragmento de vídeo medido por el tiempo de inicio y de fin.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 40 de 104



Figura 17. Ejemplo de fragmento vacío



Figura 18. Ejemplo de fragmento muerto [más de 5 segundos en la misma postura después de realizar una acción]



Figura 19. Ejemplo de acciones múltiples en un mismo video [secuencia extraída de 13 segundos de vídeo, donde al principio y final del mismo existen fragmentos muertos, además]

Es por ello que se realiza una revisión de todos los dataset en busca de estos fragmentos, y realizando recortes por medio de un *software* de edición de vídeo, los cuales se eliminan, dejando

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 41 de 104

únicamente una acción por vídeo. La duración de estas acciones varía, es decir, no se tiene un estándar de duración para todos los vídeos; sin embargo, el BOSS Dataset, al contar con varias cámaras desde donde se graban las acciones, tiene la misma duración en cada acción, pero vista desde diferente cámara, con excepción de algunas escenas donde por el ángulo de la cámara no se ve la acción, resultando en fragmentos vacíos.

3.3. Fase III

Una vez realizado el recorte, se exportan los vídeos resultantes en formatos que sean aceptados por cualquier dispositivo capaz de reproducir estas secuencias de imágenes, con el fin de garantizar una portabilidad y globalidad, es decir, que esta data resultante de la depuración pueda usarse en cualquier lugar del mundo sin excepción. Dos ejemplos de formatos globales son MP4 y AVI. De igual manera, se mantiene la resolución del vídeo original para evitar la pérdida de calidad.

Un gran problema encontrado antes de realizar los recortes de vídeo junto a su depuración era el gran tamaño de los dataset, puesto que al estar completos y contener aquellos fragmentos muertos y vacíos, extendía la duración del vídeo sin recortar.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 42 de 104

Cellphone Spanish				
Camera	Number of people	Actions/Interactions	Size	Download link
1	2	Walking/Sitting/Standing/Fighting	91MB	Cellphone_Spanish.Cam1.avi
2	2	Walking/Sitting/Standing/Fighting	114MB	Cellphone_Spanish.Cam2.avi
3	2	Walking/Sitting/Standing/Fighting	104MB	Cellphone_Spanish.Cam3.avi
4	2	Walking/Sitting/Standing/Fighting	109MB	Cellphone_Spanish.Cam4.avi
5	2	Walking/Sitting/Standing/Fighting	109MB	Cellphone_Spanish.Cam5.avi
6	2	Walking/Sitting/Standing/Fighting	113MB	Cellphone_Spanish.Cam6.avi
7	2	Walking/Sitting/Standing/Fighting	116MB	Cellphone_Spanish.Cam7.avi
8	2	Walking/Sitting/Standing/Fighting	107MB	Cellphone_Spanish.Cam8.avi
9	2	Walking/Sitting/Standing/Fighting	120MB	Cellphone_Spanish.Cam9.avi
10	1	Walking	30MB	Cellphone_Spanish.Cam10.avi

Checkout French				
Camera	Number of people	Actions/Interactions	Size	Download link
1	3	Walking/Sitting/Standing/Laying Down/Fighting	238MB	Checkout_French.Cam1.avi
2	3	Walking/Sitting/Standing/Laying Down/Fighting	280MB	Checkout_French.Cam2.avi
3	3	Walking/Sitting/Standing/Laying Down/Fighting	283MB	Checkout_French.Cam3.avi
4	3	Walking/Sitting/Standing/Laying Down/Fighting	276MB	Checkout_French.Cam4.avi
5	3	Walking/Sitting/Standing/Laying Down/Fighting	257MB	Checkout_French.Cam5.avi
6	3	Walking/Sitting/Standing/Laying Down/Fighting	280MB	Checkout_French.Cam6.avi
7	3	Walking/Sitting/Standing/Laying Down/Fighting	285MB	Checkout_French.Cam7.avi
8	3	Walking/Sitting/Standing/Laying Down/Fighting	250MB	Checkout_French.Cam8.avi
9	4	Walking/Sitting/Standing/Laying Down/Fighting	287MB	Checkout_French.Cam9.avi

Disease				
Camera	Number of people	Actions/Interactions	Size	Download link
1	3	Walking/Sitting/Standing/Laying Down	530MB	Disease.Cam1.avi
2	3	Walking/Sitting/Standing/Laying Down	603MB	Disease.Cam2.avi
3	3	Walking/Sitting/Standing/Laying Down	585MB	Disease.Cam3.avi
4	3	Walking/Sitting/Standing/Laying Down	607MB	Disease.Cam4.avi
5	3	Walking/Sitting/Standing/Laying Down	569MB	Disease.Cam5.avi
6	3	Walking/Sitting/Standing/Laying Down	608MB	Disease.Cam6.avi
7	3	Walking/Sitting/Standing/Laying Down	621MB	Disease.Cam7.avi
8	3	Walking/Sitting/Standing/Laying Down	541MB	Disease.Cam8.avi
9	5	Walking/Sitting/Standing/Laying Down	691MB	Disease.Cam9.avi

Figura 20. Vídeos del BOSS Dataset Parte 1
Tomada de: velastin.dynu.com.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 43 de 104

Disease Public				
Camera	Number of people	Actions/Interactions	Size	Download link
1	7	Walking/Sitting/Standing/Laying Down/Helping	206MB	Disease_Public.Cam1.avi
2	7	Walking/Sitting/Standing/Laying Down/Helping	232MB	Disease_Public.Cam2.avi
3	5	Walking/Sitting/Standing/Laying Down/Helping	216MB	Disease_Public.Cam3.avi
4	7	Walking/Sitting/Standing/Laying Down/Helping	237MB	Disease_Public.Cam4.avi
5	7	Walking/Sitting/Standing/Laying Down/Helping	225MB	Disease_Public.Cam5.avi
6	7	Walking/Sitting/Standing/Laying Down/Helping	235MB	Disease_Public.Cam6.avi
7	7	Walking/Sitting/Standing/Laying Down/Helping	242MB	Disease_Public.Cam7.avi
8	7	Walking/Sitting/Standing/Laying Down/Helping	212MB	Disease_Public.Cam8.avi
9	7	Walking/Sitting/Standing/Laying Down/Helping	262MB	Disease_Public.Cam9.avi

Faces 1				
Camera	Number of people	Actions/Interactions	Size	Download link
1	12	Walking	305MB	Faces_1.Cam1.avi
10	12	Walking	723MB	Faces_1.Cam10.avi

Faces 2				
Camera	Number of people	Actions/Interactions	Size	Download link
1	12	Walking	287MB	Faces_2.Cam1.avi
10	12	Walking	627MB	Faces_2.Cam10.avi

Faces 3				
Camera	Number of people	Actions/Interactions	Size	Download link
1	12	Walking	264MB	Faces_3.Cam1.avi
10	12	Walking	580MB	Faces_3.Cam10.avi

Faces 4				
Camera	Number of people	Actions/Interactions	Size	Download link
1	12	Walking	400MB	Faces_4.Cam1.avi
10	12	Walking	841MB	Faces_4.Cam10.avi

Faces 5				
Camera	Number of people	Actions/Interactions	Size	Download link
1	6	Walking	133MB	Faces_5.Cam1.avi
10	6	Walking	313MB	Faces_5.Cam10.avi

Figura 21. Vídeos del BOSS Dataset Parte 2
Tomada de: velastin.dynu.com.

DOCUMENTO FINAL

PROYECTO DE GRADO

EDO – 02

Versión 1

Vigencia:
30/06/2021

Página 44 de 104

Faces 6				
Camera	Number of people	Actions/Interactions	Size	Download link
1	13	Walking	347MB	Faces_6_Cam1.avi
10	13	Walking	756MB	Faces_6_Cam10.avi

Faces 7				
Camera	Number of people	Actions/Interactions	Size	Download link
1	11	Walking	319MB	Faces_7_Cam1.avi
10	11	Walking	697MB	Faces_7_Cam10.avi

Harass French				
Camera	Number of people	Actions/Interactions	Size	Download link
1	2	Walking/Sitting/Standing/Fighting	178MB	Harass_French.Cam1.avi
2	2	Walking/Sitting/Standing/Fighting	213MB	Harass_French.Cam2.avi
3	2	Walking/Sitting/Standing/Fighting	192MB	Harass_French.Cam3.avi
4	2	Walking/Sitting/Standing/Fighting	203MB	Harass_French.Cam4.avi
5	2	Walking/Sitting/Standing/Fighting	196MB	Harass_French.Cam5.avi
6	2	Walking/Sitting/Standing/Fighting	204MB	Harass_French.Cam6.avi
7	2	Walking/Sitting/Standing/Fighting	214MB	Harass_French.Cam7.avi
8	2	Walking/Sitting/Standing/Fighting	195MB	Harass_French.Cam8.avi
9	2	Walking/Sitting/Standing/Fighting	220MB	Harass_French.Cam9.avi
10	2	Walking	342MB	Harass_French.Cam10.avi

Harass Spanish				
Camera	Number of people	Actions/Interactions	Size	Download link
1	2	Walking/Sitting/Standing/Fighting	199MB	Harass_Spanish.Cam1.avi
2	2	Walking/Sitting/Standing/Fighting	234MB	Harass_Spanish.Cam2.avi
3	2	Walking/Sitting/Standing/Fighting	220MB	Harass_Spanish.Cam3.avi
4	2	Walking/Sitting/Standing/Fighting	226MB	Harass_Spanish.Cam4.avi
5	2	Walking/Sitting/Standing/Fighting	220MB	Harass_Spanish.Cam5.avi
6	2	Walking/Sitting/Standing/Fighting	228MB	Harass_Spanish.Cam6.avi
7	2	Walking/Sitting/Standing/Fighting	233MB	Harass_Spanish.Cam7.avi
8	2	Walking/Sitting/Standing/Fighting	213MB	Harass_Spanish.Cam8.avi
9	2	Walking/Sitting/Standing/Fighting	243MB	Harass_Spanish.Cam9.avi
10	2	Walking	403MB	Harass_Spanish.Cam10.avi

Figura 22. Vídeos del BOSS Dataset Parte 3
Tomada de: velastin.dynu.com.

Harass French 2				
Camera	Number of people	Actions/Interactions	Size	Download link
1	8	Walking/Sitting/Standing/Fighting	202MB	Harass2_French.Cam1.avi
2	8	Walking/Sitting/Standing/Fighting	246MB	Harass2_French.Cam2.avi
3	7	Walking/Sitting/Standing/Fighting	220MB	Harass2_French.Cam3.avi
4	8	Walking/Sitting/Standing/Fighting	232MB	Harass2_French.Cam4.avi
5	8	Walking/Sitting/Standing/Fighting	231MB	Harass2_French.Cam5.avi
6	8	Walking/Sitting/Standing/Fighting	234MB	Harass2_French.Cam6.avi
7	8	Walking/Sitting/Standing/Fighting	240MB	Harass2_French.Cam7.avi
8	8	Walking/Sitting/Standing/Fighting	217MB	Harass2_French.Cam8.avi
9	8	Walking	251MB	Harass2_French.Cam9.avi
10	9	Walking/Sitting/Standing/Fighting	477MB	Harass2_French.Cam10.avi

Newspaper French				
Camera	Number of people	Actions/Interactions	Size	Download link
1	6	Walking/Sitting/Standing/Fighting/Laying Down	238MB	Newspaper_French.Cam1.avi
2	6	Walking/Sitting/Standing/Fighting/Laying Down	294MB	Newspaper_French.Cam2.avi
3	5	Walking/Sitting/Standing/Fighting/Laying Down	273MB	Newspaper_French.Cam3.avi
4	6	Walking/Sitting/Standing/Fighting/Laying Down	260MB	Newspaper_French.Cam4.avi
5	6	Walking/Sitting/Standing/Fighting/Laying Down	285MB	Newspaper_French.Cam5.avi
6	6	Walking/Sitting/Standing/Fighting/Laying Down	277MB	Newspaper_French.Cam6.avi
7	6	Walking/Sitting/Standing/Fighting/Laying Down	284MB	Newspaper_French.Cam7.avi
8	6	Walking/Sitting/Standing/Fighting/Laying Down	269MB	Newspaper_French.Cam8.avi
9	6	Walking/Sitting/Standing/Fighting/Laying Down	299MB	Newspaper_French.Cam9.avi
10	6	Walking	486MB	Newspaper_French.Cam10.avi

Figura 23. Vídeos del BOSS Dataset Parte 4
Tomada de: velastin.dynu.com

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 45 de 104

Newspaper Spanish				
Camera	Number of people	Actions/Interactions	Size	Download link
1	8	Walking/Sitting/Standing/Fighting/Laying Down	157MB	Newspaper_Spanish.Cam1.avi
2	8	Walking/Sitting/Standing/Fighting/Laying Down	200MB	Newspaper_Spanish.Cam2.avi
3	7	Walking/Sitting/Standing/Fighting/Laying Down	179MB	Newspaper_Spanish.Cam3.avi
4	8	Walking/Sitting/Standing/Fighting/Laying Down	183MB	Newspaper_Spanish.Cam4.avi
5	8	Walking/Sitting/Standing/Fighting/Laying Down	191MB	Newspaper_Spanish.Cam5.avi
6	8	Walking/Sitting/Standing/Fighting/Laying Down	192MB	Newspaper_Spanish.Cam6.avi
7	8	Walking/Sitting/Standing/Fighting/Laying Down	194MB	Newspaper_Spanish.Cam7.avi
8	8	Walking/Sitting/Standing/Fighting/Laying Down	184MB	Newspaper_Spanish.Cam8.avi
9	8	Walking/Sitting/Standing/Fighting/Laying Down	208MB	Newspaper_Spanish.Cam9.avi
10	8	Walking	375MB	Newspaper_Spanish.Cam10.avi

No Event				
Camera	Number of people	Actions/Interactions	Size	Download link
1	11	Walking/Sitting/Standing/Kissing/Hugging/Handshaking	181MB	No_Event.Cam1.avi
2	11	Walking/Sitting/Standing/Kissing/Hugging/Handshaking	197MB	No_Event.Cam2.avi
3	7	Walking/Sitting/Standing/Kissing/Hugging/Handshaking	176MB	No_Event.Cam3.avi
4	11	Walking/Sitting/Standing/Kissing/Hugging/Handshaking	209MB	No_Event.Cam4.avi
5	11	Walking/Sitting/Standing/Kissing/Hugging/Handshaking	191MB	No_Event.Cam5.avi
6	11	Walking/Sitting/Standing/Kissing/Hugging/Handshaking	197MB	No_Event.Cam6.avi
7	11	Walking/Sitting/Standing/Kissing/Hugging/Handshaking	211MB	No_Event.Cam7.avi
8	11	Walking/Sitting/Standing/Kissing/Hugging/Handshaking	183MB	No_Event.Cam8.avi
9	11	Walking/Sitting/Standing/Kissing/Hugging/Handshaking	227MB	No_Event.Cam9.avi

No Event 2				
Camera	Number of people	Actions/Interactions	Size	Download link
1	9	Walking/Sitting/Standing/Handshaking/Kissing	256MB	No_Event2.Cam1.avi
2	9	Walking/Sitting/Standing/Handshaking/Kissing	327MB	No_Event2.Cam2.avi
3	8	Walking/Sitting/Standing/Handshaking/Kissing	287MB	No_Event2.Cam3.avi
4	9	Walking/Sitting/Standing/Handshaking/Kissing	301MB	No_Event2.Cam4.avi
5	9	Walking/Sitting/Standing/Handshaking/Kissing	308MB	No_Event2.Cam5.avi
6	9	Walking/Sitting/Standing/Handshaking/Kissing	310MB	No_Event2.Cam6.avi
7	9	Walking/Sitting/Standing/Handshaking/Kissing	313MB	No_Event2.Cam7.avi
8	9	Walking/Sitting/Standing/Handshaking/Kissing	297MB	No_Event2.Cam8.avi
9	9	Walking/Sitting/Standing/Handshaking/Kissing	337MB	No_Event2.Cam9.avi
10	9	Walking	620MB	No_Event2.Cam10.avi

Figura 24. Vídeos del BOSS Dataset Parte 5.
Tomada de: velastin.dynu.com

Panic				
Camera	Number of people	Actions/Interactions	Size	Download link
1	11	Walking/Sitting/Standing/Handshaking	147MB	Panic.Cam1.avi
2	11	Walking/Sitting/Standing/Handshaking	178MB	Panic.Cam2.avi
3	11	Walking/Sitting/Standing/Handshaking	178MB	Panic.Cam3.avi
4	11	Walking/Sitting/Standing/Handshaking	168MB	Panic.Cam4.avi
5	11	Walking/Sitting/Standing/Handshaking	165MB	Panic.Cam5.avi
6	11	Walking/Sitting/Standing/Handshaking	171MB	Panic.Cam6.avi
7	11	Walking/Sitting/Standing/Handshaking	172MB	Panic.Cam7.avi
8	11	Walking/Sitting/Standing/Handshaking	160MB	Panic.Cam8.avi
9	11	Walking/Sitting/Standing/Handshaking	180MB	Panic.Cam9.avi

Figura 25. Vídeos del BOSS Dataset Parte 6
Tomada de: velastin.dynu.com.

Como ejemplo, se toma el BOSS Dataset, donde su peso total es de 36264 MB o 35,41 GB (ver figuras 20, 21, 22, 23, 24 y 25), esto es, el dataset sin algún tipo de modificaciones o recortes, sino tal cual los autores lo crearon. El problema del tamaño en disco que este conjunto de datos supone en su estado puro es grave, considerando que no todas las personas cuentan con computadoras de

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 46 de 104

última generación con amplios tamaños de almacenamiento. En estos casos, se opta por realizar el proceso de descarga de algunos archivos, o su totalidad, e inmediatamente alojarlos en algún servidor en la nube, como Google Drive, para posteriormente hacer uso del Google Colaboratory. Es aquí donde surge otro problema, y es el almacenamiento que estos servidores proveen para las cuentas no institucionales, siendo de 17 GB para cuentas gratuitas; al ver el tamaño del BOSS Dataset, claramente no podría alojarse en uno de estos servidores, sin mencionar que el equipo de investigación pueda requerir más de un dataset para su proyecto.

Todo se ve desde el punto de vista gratuito, puesto que no todos los investigadores tienen los recursos para comprar espacio adicional o una cuenta institucional. En caso de realizar estos proyectos de manera independiente, es importante ver el impacto económico que supone todo este proceso.

Ahora bien, suponiendo que el o los dataset usados por el grupo de investigación X ocupe un tamaño igual o inferior al ofrecido por un servidor gratuito en la nube, tomando como referencia a América Latina, la mayoría de su población accede a internet con una baja calidad, teniendo velocidades de bajada y subida bastante bajas, lo que representa un retraso en tiempo para el grupo de investigación al momento de cargar en la nube el o los conjuntos de datos a usar.

Por otra parte, continuando con el BOSS Dataset como ejemplo, se evidencia una limitación de velocidad de descarga por parte del servidor donde se encuentra alojado, impidiendo aprovechar todo el ancho de banda que se pueda tener, y teniendo una limitación de banda aproximada de 320 KB/s (ver Figura 26), haciendo que una descarga de 530 MB consuma una cantidad de tiempo considerable. Si se descargase el dataset completo, supondría un gran gasto de tiempo.

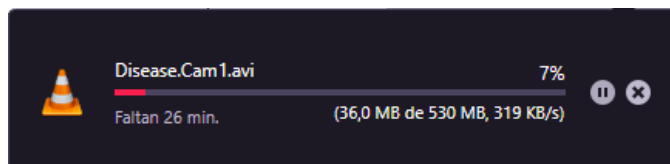


Figura 26. Tiempo de descarga desde el servidor que está alojado el BOSS Dataset

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 47 de 104

Al tener el RFDD alojado en un servidor que no limite el ancho de banda al momento de la descarga, el equipo de investigación ahorra gran cantidad de tiempo, pudiendo usar toda la capacidad que su conexión a internet le ofrezca.

La optimización no solo es en cuestión de tiempo de carga o descarga, sino en el tamaño de los vídeos. Se realiza la comparación entre el peso original del BOSS Dataset y el peso de los vídeos resultantes del recorte y depuración de los mismos (ver Figura 28) y observando una optimización de tamaño de aproximadamente un 96,38 % (ver Figura 27), reduciendo drásticamente la carga computacional en el procesamiento de la data.

Siendo un dataset más liviano después de pasar por el proceso de depuración y recorte, es más sencillo optar por trabajar directamente en la nube, al dejar de lado la enorme cantidad de tiempo que tomaría cargar a la nube un dataset tan pesado como lo es el original.

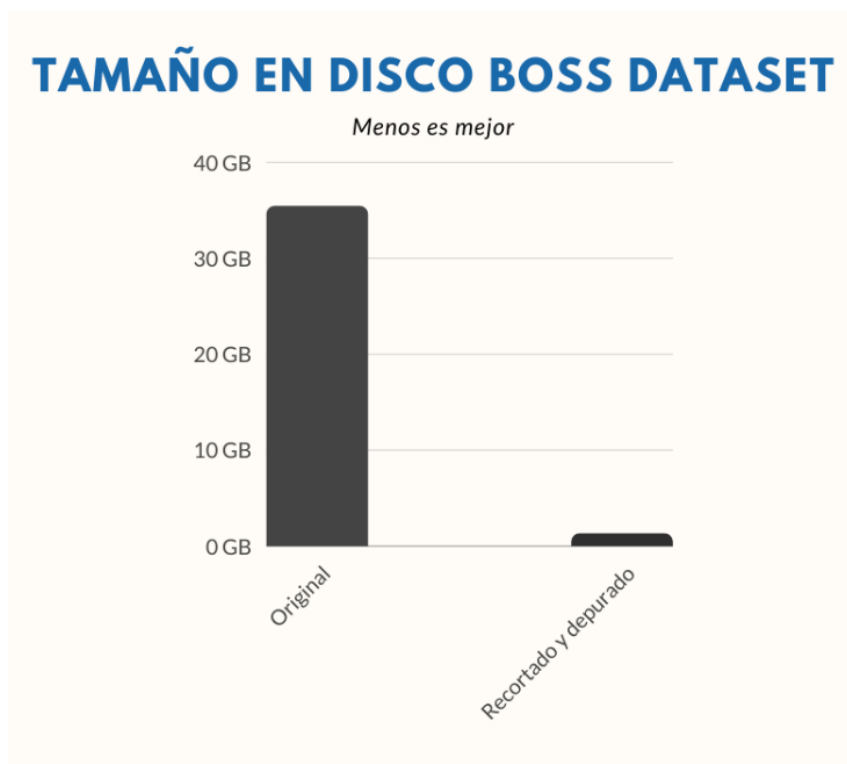


Figura 27. Optimización de tamaño BOSS Dataset

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 48 de 104

General Detalles 1.278 archivos, 0 carpetas Tipo: Todos del tipo MP4 Video File (VLC) Ubicación: Todos en D:\Dataset\RFDD\ADL\RGB Tamaño: 1,24 GB (1.334.094.800 bytes) Tamaño en disco: 1,24 GB (1.336.709.120 bytes)	General Detalles 39 archivos, 0 carpetas Tipo: Todos del tipo MP4 Video File (VLC) Ubicación: Todos en D:\Dataset\RFDD\FALL\RGB Tamaño: 41,6 MB (43.711.833 bytes) Tamaño en disco: 41,7 MB (43.782.144 bytes)
--	--

Figura 28. Peso de los archivos resultantes del BOSS Dataset después de recorte y depuración.

3.4. Fase IV

Teniendo los vídeos recortados, se organizan en dos clases: ADL y FALL, esto con el fin de que sea más sencillo realizar el procesamiento de los vídeos. Además, se generan dos listas con los nombres de todos los vídeos que conforman el nuevo dataset: una para identificar el directorio respectivo de cada vídeo, sea ADL o FALL, y otra donde se añade al final de cada nombre de vídeo la clase correspondiente (ADL – 1, FALL – 2) a la que pertenece. DE esta forma, la red neuronal podrá identificar qué acciones son ADL y cuáles caídas.

Se procede, por medio de un algoritmo, a extraer muestras cada cierta cantidad de *frames* de cada vídeo, tanto ADL como FALL, y almacenarlas en un directorio reservado solo para estas muestras. Un vídeo, al ser una secuencia de imágenes, es sencillo extraer diferentes de estas.

Como ejemplo, para el primer vídeo del dataset en la clase ADL, el algoritmo extrae 3 *frames* del vídeo original (ver Figura 29), cuya duración de la acción que contiene es de 2 segundos, mientras que para el vídeo 27 de ADL del dataset, cuya duración es de 4 segundos, el algoritmo extrae más *frames* (5) (ver Figura 30). El total de *frames* extraídos entre todos los vídeos de ADL y FALL resultantes de los recortes previos a todos los dataset usados en el proyecto fue de 13 955 (ver Figura 31).

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 49 de 104

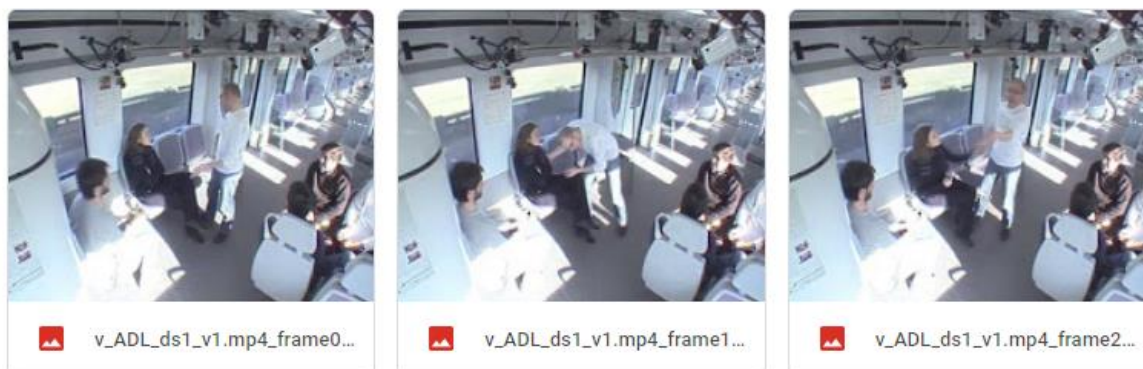


Figura 29. Frames extraídos del primer video del dataset – Clase ADL

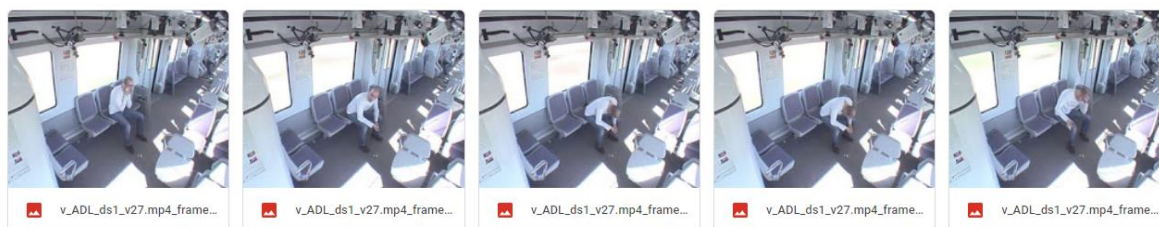


Figura 30. Frames extraídos del 27vo video del dataset – Clase ADL

100% | 13955/13955 [00:00<00:00, 453883.95it/s]

Figura 31. Cantidad de frames extraídos de todos los vídeos

Se estructura todos estos *frames* en un archivo CSV, que contendrá, entonces, los nombres de las muestras extraídas y sus respectivas clases (ADL o FALL) (ver Figura 32), y, de esta forma, en el proceso de entrenamiento poder cargar con mayor facilidad las muestras extraídas de los vídeos, pero para ello se debe realizar un ajuste en la data estructurada, ya que se debe trabajar con la misma cantidad de vídeos del dataset.

Como el RFDD consta de 1925 (ver Figura 33) vídeos comprendidos entre las clases ADL y FALL, se debe ajustar la cantidad de registros del CSV a la cantidad de vídeos de cada clase,

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 50 de 104

eliminando algunos registros de *frames*. Hacer esto no supone problema para el entrenamiento, ya que con la misma cantidad de vídeos a trabajar es suficiente. De esta manera, se garantiza que el algoritmo haga un emparejamiento de manera correcta entre la data estructurada y la no estructurada (ver Figura 34).

A	B	Column1	Column2
1	Column1	Column2	
2	image	class	
3	v_ADL_ds1_v1.mp4_frame0.jpg	ADL	
4	v_ADL_ds1_v1.mp4_frame1.jpg	ADL	
5	v_ADL_ds1_v1.mp4_frame2.jpg	ADL	
6	v_ADL_ds1_v2.mp4_frame0.jpg	ADL	
7	v_ADL_ds1_v2.mp4_frame1.jpg	ADL	
8	v_ADL_ds1_v3.mp4_frame0.jpg	ADL	
9	v_ADL_ds1_v3.mp4_frame1.jpg	ADL	
10	v_ADL_ds1_v4.mp4_frame0.jpg	ADL	
11	v_ADL_ds1_v4.mp4_frame1.jpg	ADL	
12	v_ADL_ds1_v5.mp4_frame0.jpg	ADL	
13	v_ADL_ds1_v5.mp4_frame1.jpg	ADL	
14	v_ADL_ds1_v6.mp4_frame0.jpg	ADL	
15	v_ADL_ds1_v6.mp4_frame1.jpg	ADL	
16	v_ADL_ds1_v7.mp4_frame0.jpg	ADL	
17	v_ADL_ds1_v7.mp4_frame1.jpg	ADL	
18	v_ADL_ds1_v8.mp4_frame0.jpg	ADL	
19	v_ADL_ds1_v8.mp4_frame1.jpg	ADL	
20	v_ADL_ds1_v8.mp4_frame2.jpg	ADL	
21	v_ADL_ds1_v9.mp4_frame0.jpg	ADL	
13940	v_FALL_ds7_v57.mp4_frame0.jpg	FALL	
13941	v_FALL_ds7_v57.mp4_frame1.jpg	FALL	
13942	v_FALL_ds7_v57.mp4_frame2.jpg	FALL	
13943	v_FALL_ds7_v57.mp4_frame3.jpg	FALL	
13944	v_FALL_ds7_v57.mp4_frame4.jpg	FALL	
13945	v_FALL_ds7_v58.mp4_frame0.jpg	FALL	
13946	v_FALL_ds7_v58.mp4_frame1.jpg	FALL	
13947	v_FALL_ds7_v58.mp4_frame2.jpg	FALL	
13948	v_FALL_ds7_v58.mp4_frame3.jpg	FALL	
13949	v_FALL_ds7_v58.mp4_frame4.jpg	FALL	
13950	v_FALL_ds7_v59.mp4_frame0.jpg	FALL	
13951	v_FALL_ds7_v59.mp4_frame1.jpg	FALL	
13952	v_FALL_ds7_v59.mp4_frame2.jpg	FALL	
13953	v_FALL_ds7_v59.mp4_frame3.jpg	FALL	
13954	v_FALL_ds7_v60.mp4_frame0.jpg	FALL	
13955	v_FALL_ds7_v60.mp4_frame1.jpg	FALL	
13956	v_FALL_ds7_v60.mp4_frame2.jpg	FALL	
13957	v_FALL_ds7_v60.mp4_frame3.jpg	FALL	

Figura 32. CSV Frames generados

100% | 1925/1925 [15:59<00:00, 2.01it/s]

Figura 33. Cantidad de vídeos en el RFDD

A	B	Column1	Column2
1	Column1	Column2	
2	image	class	
3	v_ADL_ds1_v1.mp4_frame0.jpg	ADL	
4	v_ADL_ds1_v1.mp4_frame1.jpg	ADL	
5	v_ADL_ds1_v1.mp4_frame2.jpg	ADL	
6	v_ADL_ds1_v2.mp4_frame0.jpg	ADL	
7	v_ADL_ds1_v2.mp4_frame1.jpg	ADL	
8	v_ADL_ds1_v3.mp4_frame0.jpg	ADL	
9	v_ADL_ds1_v3.mp4_frame1.jpg	ADL	
10	v_ADL_ds1_v4.mp4_frame0.jpg	ADL	
11	v_ADL_ds1_v4.mp4_frame1.jpg	ADL	
12	v_ADL_ds1_v5.mp4_frame0.jpg	ADL	
13	v_ADL_ds1_v5.mp4_frame1.jpg	ADL	
14	v_ADL_ds1_v6.mp4_frame0.jpg	ADL	
15	v_ADL_ds1_v6.mp4_frame1.jpg	ADL	
1913	v_FALL_ds7_v5.mp4_frame5.jpg	FALL	
1914	v_FALL_ds7_v5.mp4_frame6.jpg	FALL	
1915	v_FALL_ds7_v5.mp4_frame7.jpg	FALL	
1916	v_FALL_ds7_v5.mp4_frame8.jpg	FALL	
1917	v_FALL_ds7_v5.mp4_frame9.jpg	FALL	
1918	v_FALL_ds7_v5.mp4_frame10.jpg	FALL	
1919	v_FALL_ds7_v6.mp4_frame0.jpg	FALL	
1920	v_FALL_ds7_v6.mp4_frame1.jpg	FALL	
1921	v_FALL_ds7_v6.mp4_frame2.jpg	FALL	
1922	v_FALL_ds7_v6.mp4_frame3.jpg	FALL	
1923	v_FALL_ds7_v6.mp4_frame4.jpg	FALL	
1924	v_FALL_ds7_v6.mp4_frame5.jpg	FALL	
1925	v_FALL_ds7_v6.mp4_frame6.jpg	FALL	
1926	v_FALL_ds7_v6.mp4_frame7.jpg	FALL	
1927	v_FALL_ds7_v16.mp4_frame0.jpg	FALL	

Figura 34. CSV Ajustado a la cantidad de vídeos del RFDD

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 51 de 104

Luego de ajustar la data estructurada, se procede a cargar dicha data y entrenar la red neuronal con las diferentes arquitecturas de red usadas para este proyecto, guardando sus pesos. tanto con el nombre de la arquitectura como la hora específica de entrenamiento (ver Figura 35).

Este entrenamiento se realizó para cada arquitectura con 100 épocas. El código de entrenamiento está diseñado para cambiar de arquitectura de red tan solo con enviar como parámetro el nombre de la arquitectura (ver Figura 36).

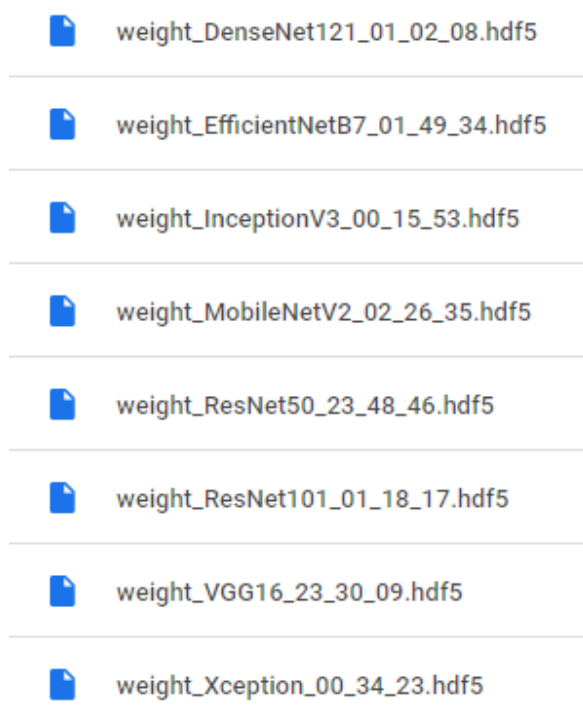


Figura 35. Pesos de arquitecturas de red después del entrenamiento

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 52 de 104

Las líneas 22 y 23 son usadas para obtener la hora actual y ajustarlo al formato de horas-minutos-segundos que posteriormente se agrega al archivo de pesos de la arquitectura de red convolucional correspondiente, esto con el fin de llevar un registro en tiempo para cada entrenamiento. La línea 26 importa la función para almacenar los pesos de la arquitectura de red neuronal convolucional en entrenamiento. La línea 28 cumple el papel de parámetro, es donde se define la arquitectura de red convolucional a usar de las disponibles en el diccionario (líneas 31 a 40). La línea 29 define la cantidad de épocas a entrenar. Las líneas 42 lee el archivo csv generado anteriormente y la línea 43 muestra en pantalla los primeros 5 registros. Por último, la línea 112 almacena los pesos del entrenamiento de la arquitectura de red convolucional usada en un directorio establecido añadiendo al nombre del archivo la hora actual.

```

-> VGG16
-> ResNet50
-> InceptionV3
-> Xception
-> DenseNet121
-> ResNet101
-> EfficientNetB7
-> MobileNetV2

22 t = time.localtime()
23 current_time = time.strftime("%H_%M_%S", t)
24
25 # defining a function to save the weights of best model
26 from keras.callbacks import ModelCheckpoint
27
28 choice_of_archs = "VGG16"
29 epochs = 100
30
31 architectures = {
32     "VGG16": VGG16(weights='imagenet', include_top=False),
33     "ResNet50": ResNet50(weights='imagenet', include_top=False),
34     "InceptionV3": InceptionV3(weights='imagenet', include_top=False),
35     "Xception": Xception(weights='imagenet', include_top=False),
36     "DenseNet121": DenseNet121(weights='imagenet', include_top=False),
37     "ResNet101": ResNet101(weights='imagenet', include_top=False),
38     "EfficientNetB7": EfficientNetB7(weights='imagenet', include_top=False),
39     "MobileNetV2": MobileNetV2(weights='imagenet', include_top=False)
40 }
41
42 train = pd.read_csv('/content/drive/MyDrive/RobustFallDetection_Dataset_Copia/RGB/train_new2.csv')
43 train.head()

111 #Save corresponding architecture weight + current time
112 filename_weight = '/content/drive/MyDrive/RobustFallDetection_Dataset_Copia/Weights/weight_'+choice_of_archs+'_'+current_time+'.hdf5'

```

Figura 36. Entrenamiento de arquitecturas de red con RFDD

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 53 de 104

3.5. Fase V

Después de realizar el entrenamiento del **RFDD** con las diferentes arquitecturas de red neuronal, se **realizó** la validación cargando los pesos de cada arquitectura según corresponda (ver Figura 37); adicional, se especifica al algoritmo la última capa de la arquitectura (ver Figura 38), ya que **fueron** los valores de esta los que se **usaron** como entrada en el modelo de arquitectura. Estos códigos serán explicados en detalle en el siguiente capítulo.

```
path_to_weights = "/content/drive/MyDrive/RobustFallDetection_Dataset_Copia/Weights/weight_EfficientNetB7_01_49_34.hdf5"
choice_of_archs = "EfficientNetB7"
```

Figura 37. Carga de pesos con arquitectura de red neuronal correspondiente

<pre>base_model = architectures[choice_of_archs] t = base_model.get_layer('top_activation').output.get_shape() InputShape = t[1]*t[2]*t[3] print(InputShape) #defining the model architecture model = Sequential() model.add(Dense(1024, activation='relu', input_shape=(InputShape,))) model.add(Dropout(0.5)) model.add(Dense(512, activation='relu')) model.add(Dropout(0.5)) model.add(Dense(256, activation='relu')) model.add(Dropout(0.5)) model.add(Dense(128, activation='relu')) model.add(Dropout(0.5)) model.add(Dense(2, activation='softmax'))</pre> EfficientNetB7	<pre>base_model = architectures[choice_of_archs] t = base_model.get_layer('block14_sepconv2_act').output.get_shape() InputShape = t[1]*t[2]*t[3] print(InputShape) #defining the model architecture model = Sequential() model.add(Dense(1024, activation='relu', input_shape=(InputShape,))) model.add(Dropout(0.5)) model.add(Dense(512, activation='relu')) model.add(Dropout(0.5)) model.add(Dense(256, activation='relu')) model.add(Dropout(0.5)) model.add(Dense(128, activation='relu')) model.add(Dropout(0.5)) model.add(Dense(2, activation='softmax'))</pre> Xception
---	---


```
base_model = architectures[choice_of_archs]
t = base_model.get_layer('out_relu').output.get_shape()
InputShape = t[1]*t[2]*t[3]

print(InputShape)

#defining the model architecture
model = Sequential()
model.add(Dense(1024, activation='relu', input_shape=(InputShape,)))
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))
```

MobileNetV2

Figura 38. Ejemplos de las diferentes capas finales de arquitecturas de red

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 54 de 104

Cabe resaltar que, a pesar de haber realizado el recorte de los vídeos, y con ello optimizado la carga computacional que supone el procesamiento de todos estos vídeos y sus respectivas muestras, incluso al usar Google Colaboratory, se evidenciaron altos tiempos de procesamiento para algunas arquitecturas (ver Figura 39). Estos procesos, tanto de entrenamiento como de *validación*, se realizaron usando el procesador que Google Colaboratory ofrece. La no realización del proceso con el uso de GPU dedicada es debido a que muchas personas no cuentan con tarjetas de vídeo Nvidia, ya que son aquellas tarjetas las que funcionan con tensorflow, y que realizan estas tareas en menores tiempos, mientras que todas las computadoras poseen procesadores que funcionan correctamente, independiente de la marca; claramente, algunos tendrán mejores resultados que otros.

Es de suponer entonces, que, de realizar todo el proceso con los vídeos originales, la carga computacional aumentaría de manera considerable, ya que las secuencias de imágenes al principio eran más extensas y con fragmentos que hubiesen entorpecido el entrenamiento de la red neuronal, y los resultados obtenidos al final de este proceso serían mediocres y poco prácticos.

En el caso de trabajar con el RFDD de manera local, se requiere una computadora con bastantes recursos de *hardware* capaces de realizar el entrenamiento y *validación* en menor tiempo que, por ejemplo, Google Colaboratory.

```

100%|██████████| 1925/1925 [1:46:11<00:00, 3.31s/it] -> VGG-16
100%|██████████| 1925/1925 [1:05:13<00:00, 2.03s/it] -> ResNet50
100%|██████████| 1925/1925 [44:37<00:00, 1.39s/it] -> InceptionV3
100%|██████████| 1925/1925 [1:04:51<00:00, 2.02s/it] -> Xception
100%|██████████| 1925/1925 [1:03:51<00:00, 1.99s/it] -> DenseNet121
100%|██████████| 1925/1925 [1:49:04<00:00, 3.40s/it] -> ResNet101
100%|██████████| 1925/1925 [2:08:33<00:00, 4.01s/it] -> EfficientNetB7
100%|██████████| 1925/1925 [46:57<00:00, 1.46s/it] -> MobileNetV2

```

Figura 39. Tiempos de validación de arquitecturas de redes neuronales convolucionales

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 55 de 104

Para la creación del dataset fue necesario recopilar información de investigaciones relacionadas con las caídas de personas. Estas investigaciones, a pesar de que contaban con dataset funcionales, tenían falencias en cuanto a la optimización en las secuencias de imágenes. Cabe resaltar que toda la data de estos conjuntos de datos era de vídeo; en el mejor de los casos, estos vídeos tenían “fragmentos muertos”, dado que, tras haber pasado la acción, fuere de caída o ADL, el vídeo continuaba estático mostrando una secuencia de *frames* de la acción finalizada, resultando en una pérdida de recursos computacionales a la hora de realizar el entrenamiento al tener más información que la necesaria para ser procesada.

En el peor de los casos, había muchos fragmentos de vídeo donde no había acción alguna, generando al final una secuencia de imágenes de bastante duración y gran tamaño que, nuevamente, resulta inútil para el entrenamiento y validación, además de retrasar el proceso.

Para la construcción del conjunto de datos robusto en esta investigación fueron descargados 7 conjuntos de datos públicos relacionados con acciones de caídas y del diario vivir.

La tabla 1, describe por cada conjunto de datos: nombre, año de publicación, cantidad de videos relacionados a las dos clases de interés para el proyecto: ADL y Fall. Por otra parte, presenta una muestra del contenido del video y se indica si el conjunto de datos original fue editado para ser incluido como resultado de esta investigación removiendo los fragmentos residuales y vacíos.

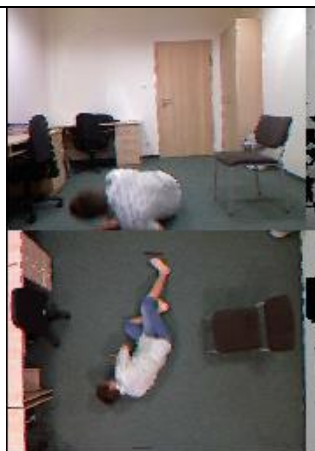
	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 56 de 104

Tabla 1. Datasets públicos usados en el proyecto

Conjunto de datos	Año	Videos resultantes	Muestra de video	Recorte
BOSS Dataset	2017	ADL: 1278 FALL: 39		Si
Cogent Labs Dataset	2015	ADL: 188 FALL: 71		Si
Evaluating Depth Dataset	2014	ADL: 0 Fall: 149		No
Fall Detection Dataset	2017	ADL: 79 FALL: 8		No
SisFall Dataset	2017	ADL: 25 FALL: 15		Si

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 57 de 104

Continuación Tabla 1, conjuntos de datos descargados en el proyecto.

Conjunto de datos	Año	Videos resultantes	Muestra de video	Recorte
UR Fall Detection Dataset	2014	ADL: 0 FALL: 60		No
UMA Fall Dataset	2016	ADL: 10 FALL: 3		Si

La tabla 1, describe por cada conjunto de datos: nombre, año de publicación, cantidad de videos relacionados a las dos clases de interés para el proyecto: ADL y Fall. Por otra parte, presenta una muestra del contenido del video y se indica si el conjunto de datos original fue editado para ser incluido como resultado de esta investigación removiendo los fragmentos residuales y vacíos.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 58 de 104

CAPÍTULO 4 - ALGORITMO

4.1. ALGORITMO DE RED NEURONAL

Para el presente proyecto, se tomó como base el algoritmo propuesto por Sharma, Pulkit [64], donde desarrolla un modelo de clasificación de vídeo capaz de clasificarlos en su categoría o clase correspondiente. El modelo se divide en los siguientes pasos:

1. Crear los archivos de entrenamiento y pruebas que contienen el directorio donde se encuentran y los nombres de los archivos para el primer archivo (ver Figura 40), y para el segundo, de igual manera los nombres de los archivos, pero adicionando su clase correspondiente (ver Figura 41); adicional, una jerarquía de directorios donde se almacenarán los vídeos que componen el dataset, el archivo de estructuración de datos en formato csv, los dos archivos de texto con los nombres de los vídeos y sus clases, *frames* de muestra temporales, pesos de las diferentes arquitecturas de red y muestras de cada vídeo del dataset (ver Figura 42).

```
ADL/v_ADL_ds1_v1.mp4
ADL/v_ADL_ds1_v2.mp4
ADL/v_ADL_ds1_v3.mp4
ADL/v_ADL_ds1_v4.mp4
ADL/v_ADL_ds1_v5.mp4
ADL/v_ADL_ds1_v6.mp4
FALL/v_FALL_ds1_v4.mp4
FALL/v_FALL_ds1_v5.mp4
FALL/v_FALL_ds1_v6.mp4
FALL/v_FALL_ds1_v7.mp4
FALL/v_FALL_ds1_v8.mp4
FALL/v_FALL_ds1_v9.mp4
```

Figura 40. Archivo de entrenamiento (trainlist01.txt)

```
ADL/v_ADL_ds1_v1.mp4 1
ADL/v_ADL_ds1_v2.mp4 1
ADL/v_ADL_ds1_v3.mp4 1
ADL/v_ADL_ds1_v4.mp4 1
ADL/v_ADL_ds1_v5.mp4 1
ADL/v_ADL_ds1_v6.mp4 1
ADL/v_ADL_ds1_v7.mp4 1
ADL/v_ADL_ds1_v8.mp4 1
ADL/v_ADL_ds1_v9.mp4 1
ADL/v_ADL_ds5_v10.mp4 1
FALL/v_FALL_ds1_v1.mp4 2
FALL/v_FALL_ds1_v2.mp4 2
FALL/v_FALL_ds1_v3.mp4 2
FALL/v_FALL_ds1_v4.mp4 2
FALL/v_FALL_ds1_v5.mp4 2
FALL/v_FALL_ds1_v6.mp4 2
FALL/v_FALL_ds1_v7.mp4 2
FALL/v_FALL_ds1_v8.mp4 2
FALL/v_FALL_ds1_v9.mp4 2
```

Figura 41. Archivo de prueba (testlist01.txt)

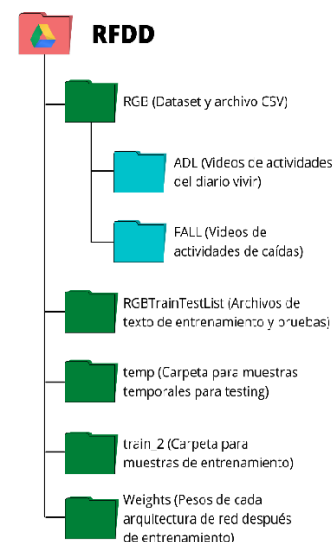


Figura 42. Árbol de directorios para el proyecto

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 59 de 104

2. Extraer las muestras de todos los vídeos listados en el archivo de entrenamiento y el de prueba. Se abre cada archivo y se crea un dataframe con los nombres de los vídeos (ver Figura 43): Desde la línea no. 1 hasta la no. 11 se importan las librerías necesarias, se abre el archivo de lista de entrenamiento en la línea 13 y en la línea 14 se lee dicho archivo, la línea 15 separa cada línea de texto del archivo y de esta forma separa cada vídeo. De la línea no. 18 a la no. 21 se crea el dataframe que contendrá los nombres de los vídeos pasando como parámetro la información de la variable videos de la línea no. 15, por último, se imprimen los 5 primeros registros del dataframe. Las líneas 24, 25 y 26 realizan la misma función que las líneas 13, 14 y 15 cambiando únicamente el archivo que se abre y lee, en las líneas anteriores se trabajaba con el archivo de entrenamiento, ahora se usa el archivo de validación y de nuevo se crea un dataframe para los videos de este archivo desde la línea 29 a 32 realizando el mismo proceso que las líneas 18, 19, 20 y 21.

```

1  import cv2
2  import math
3  import matplotlib.pyplot as plt
4  import pandas as pd
5  from keras.preprocessing import image
6  import numpy as np
7  from keras.utils import np_utils
8  from skimage.transform import resize
9  from sklearn.model_selection import train_test_split
10 from glob import glob
11 from tqdm import tqdm
12
13 f = open("/content/drive/MyDrive/RobustFallDetection_Dataset_Copia/RGBTrainTestList/trainlist01.txt", "r")
14 temp = f.read()
15 videos = temp.split('\n')
16
17 # creating a dataframe having video names
18 train = pd.DataFrame()
19 train['video_name'] = videos
20 train = train[:-1]
21 print(train.head())
22
23 # open the .txt file which have names of test videos
24 f = open("/content/drive/MyDrive/RobustFallDetection_Dataset_Copia/RGBTrainTestList/testlist01.txt", "r")
25 temp = f.read()
26 videos = temp.split('\n')
27
28 # creating a dataframe having video names
29 test = pd.DataFrame()
30 test['video_name'] = videos
31 test = test[:-1]
32 print(test.head())

```

Figura 43. Lectura de la lista de videos en los archivos de entrenamiento y validación

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 60 de 104

Seguidamente, se crean las etiquetas para los vídeos de entrenamiento y validación (ver Figuras 44 y 45): se define un arreglo que almacenará las etiquetas de las clases (ADL y FALL) en la línea 35 y en las líneas 36 y 37 se realiza un bucle for con el objetivo de extraer dichas etiquetas desde el dataframe de entrenamiento previamente creado, en la línea 39 se definen las etiquetas extraídas para la columna ‘tag’ del dataframe. Las líneas 43, 44, 45 y 47 realizan el mismo procedimiento para el dataframe de validación. Se procede con la extracción de muestras, etiquetándola con su debida clase y almacenándolas en la carpeta de muestras train_2 (ver figuras 46 y 47): se define otro ciclo for en la línea no. 50 que servirá para extraer frames de los vídeos contenidos en el dataset, la línea 52 provee el nombre de cada video a leer. La línea 56 abre el vídeo indicado en la línea 52 en el directorio especificado, en la siguiente línea (57) se establece el *frame rate* deseado, en este caso es de 5. Se define un ciclo *while* en la línea 59 para que se extraigan frames de cada video mientras el capturador de vídeo de la línea 56 permanezca activo; las líneas 60 - 67 extraen los frames y almacenan en un directorio definido, una vez extraídos todos los frames permitidos por cada video, se libera la capturadora de video en la línea 68.

```

34 # creating tags for training videos
35 train_video_tag = []
36 for i in range(train.shape[0]):
37     train_video_tag.append(train['video_name'][i].split('/')[0])
38
39 train['tag'] = train_video_tag
40 print(train)
41
42 # creating tags for test videos
43 test_video_tag = []
44 for i in range(test.shape[0]):
45     test_video_tag.append(test['video_name'][i].split('/')[0])
46
47 test['tag'] = test_video_tag

```

Figura 44. Creación de etiquetas para vídeos de entrenamiento y prueba

	video_name	tag
0	ADL/v_ADL_ds1_v1.mp4	1 ADL
1	ADL/v_ADL_ds1_v2.mp4	1 ADL
2	ADL/v_ADL_ds1_v3.mp4	1 ADL
3	ADL/v_ADL_ds1_v4.mp4	1 ADL
4	ADL/v_ADL_ds1_v5.mp4	1 ADL
...	...	...
1920	FALL/v_FALL_ds7_v56.mp4	2 FALL
1921	FALL/v_FALL_ds7_v57.mp4	2 FALL
1922	FALL/v_FALL_ds7_v58.mp4	2 FALL
1923	FALL/v_FALL_ds7_v59.mp4	2 FALL
1924	FALL/v_FALL_ds7_v60.mp4	2 FALL

Figura 45. Nombres de video con su respectiva etiqueta

```

49 # storing the frames from training videos
50 for i in tqdm(range(train.shape[0])):
51     count = 0
52     videoFile = train['video_name'][i]
53     print(videoFile)
54
55     # capturing the video from the given path
56     cap = cv2.VideoCapture('/content/drive/MyDrive/RobustFallDetection_Dataset_Copia/RGB/'+train['tag'][i]+'/' + videoFile.split('/')[0].split('/')[1])
57     frameRate = cap.get(5) #frame rate
58     x=1
59     while(cap.isOpened()):
60         frameId = cap.get(1) #current frame number
61         ret, frame = cap.read()
62         if (ret != True):
63             break
64         if (frameId % math.floor(frameRate) == 0):
65             # storing the frames in a new folder named train_2
66             filename = '/content/drive/MyDrive/RobustFallDetection_Dataset_Copia/train_2/' + videoFile.split('/')[1].split('/')[0] + "_frame%d.jpg" % count; count+=1
67             cv2.imwrite(filename, frame)
68     cap.release()

```

Figura 46. Extracción de muestras del dataset

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 61 de 104

La red neuronal usará estas muestras para realizar su debido entrenamiento, identificando *frame* a *frame* la acción que está digitalizada; de esta forma, una vez realizado todo el proceso, la red será capaz de identificar acciones en vídeos ajenos a los que componen el RFDD.

```
[1925 rows x 2 columns]
ADL/v_AD_L_ds1_v1.mp4 1
0%|          | 1/1925 [00:00<07:33, 4.25it/s]ADL/v_AD_L_ds1_v2.mp4 1
0%|          | 2/1925 [00:00<07:25, 4.31it/s]
0%|          | 3/1925 [00:00<07:11, 4.46it/s]ADL/v_AD_L_ds1_v3.mp4 1
0%|          | 4/1925 [00:00<06:45, 4.73it/s]ADL/v_AD_L_ds1_v4.mp4 1
ADL/v_AD_L_ds1_v5.mp4 1
0%|          | 5/1925 [00:01<06:33, 4.88it/s]
0%|          | 6/1925 [00:01<06:13, 5.13it/s]ADL/v_AD_L_ds1_v6.mp4 1
ADL/v_AD_L_ds1_v7.mp4 1
0%|          | 7/1925 [00:01<06:09, 5.19it/s]ADL/v_AD_L_ds1_v8.mp4 1
100%|██████████| 1920/1925 [15:57<00:01, 3.13it/s]FALL/v_FALL_ds7_v56.mp4 2
100%|██████████| 1921/1925 [15:58<00:01, 3.23it/s]FALL/v_FALL_ds7_v57.mp4 2
100%|██████████| 1922/1925 [15:58<00:00, 3.08it/s]FALL/v_FALL_ds7_v58.mp4 2
100%|██████████| 1923/1925 [15:59<00:00, 2.91it/s]FALL/v_FALL_ds7_v59.mp4 2
100%|██████████| 1924/1925 [15:59<00:00, 3.06it/s]FALL/v_FALL_ds7_v60.mp4 2
100%|██████████| 1925/1925 [15:59<00:00, 2.01it/s]
```

Figura 47. Proceso de extracción de muestras completado correctamente

3. Estructurar la data de forma que las muestras de los vídeos se encuentren ordenadas junto a su clase correspondiente. Este código obtiene el nombre de todas las imágenes, accediendo al directorio `train_2`, almacenándolos en un array al igual que su clase correspondiente y luego almacenando las imágenes y sus clases en un dataframe con sus respectivos títulos “*image*” y “*class*”; por último, este dataframe se convierte en un archivo csv, generando así un listado con el nombre de cada imagen y su clase (ver Figura 48): Las 4 primeras líneas importan las librerías necesarias, la línea 7 obtiene los nombres de todos los frames almacenados en la carpeta `train_2`. Se definen dos arreglos, uno para la imagen de entrenamiento y otro para la etiqueta (ADL o

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 62 de 104

FALL) de esta forma se podrá asociar cada acción contenida en un frame, a una etiqueta específica. Un ciclo for es definido desde la línea 13 hasta la línea 18 para almacenar los nombres de cada frame y su respectiva etiqueta en los arreglos creados en las líneas 11 y 12. Ahora estas imágenes y sus clases correspondientes se almacenan en un dataframe de entrenamiento de tal manera que quedarán relacionadas. Por último, en la línea 26, este dataframe se convierte en un archivo csv. En este punto, el dataframe actúa sobre todas las muestras extraídas que se encuentran alojadas en el directorio train_2, es por esto que se debe realizar un ajuste de cantidad de acuerdo a la cantidad de vídeos del dataset RFDD (ver figuras 32, 33 y 34) y así tener concordancia en cuanto a las cantidades a trabajar; de lo contrario, se generará un error de lectura en los archivos.

```

1  import numpy as np
2  from glob import glob
3  from tqdm import tqdm
4  import pandas as pd
5
6  # getting the names of all the images
7  images = glob("/content/drive/MyDrive/RobustFallDetection_Dataset_Copia/train_2/*.jpg")
8
9  print(images)
10
11  train_image = []
12  train_class = []
13  for i in tqdm(range(len(images))):
14      # creating the image name
15      train_image.append(images[i].split('/')[6])
16
17      # creating the class of image
18      train_class.append(images[i].split('/')[6].split('_')[1])
19
20  # storing the images and their class in a dataframe
21  train_data = pd.DataFrame()
22  train_data['image'] = train_image
23  train_data['class'] = train_class
24
25  # converting the dataframe into csv file
26  train_data.to_csv('/content/drive/MyDrive/RobustFallDetection_Dataset_Copia/RGB/train_new2.csv',header=True, index=False)

```

Figura 48. Creación del archivo csv

4. Se realiza el entrenamiento creando una función para capturar la hora actual al momento de realizar el entrenamiento, para luego anexarla al archivo de peso de la arquitectura a trabajar. Para esto, se define la arquitectura de red a usar y la cantidad de épocas, se crea un diccionario de arquitecturas para optimizar la elección de las mismas, de tal manera que únicamente se pase como

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 63 de 104

parámetro el nombre de la arquitectura deseada para trabajar (ver Figura 36), y luego se procede con los siguientes pasos:

4.1. En la figura 49, las líneas 1 - 20 importan las librerías necesarias. Se captura la hora actual en 22 y se transforma al formato de hora-minutos-segundos en 23. Se define una función para almacenar los pesos de la arquitectura de red neuronal convolucional a usar en 26. La línea 28 servirá de parámetro para especificar la arquitectura de red neuronal convolucional a usar en el entrenamiento, el nombre de la arquitectura se extrae del diccionario definido en las líneas 31 – 40, la línea 29 define la cantidad de épocas a entrenar.

En la línea 42 se abre el archivo csv creado previamente y se crea un array vacío para luego almacenar los frames. Se define un ciclo for en la línea 49 para recorrer el archivo csv, en 51 se carga la imagen correspondiente al registro actual del archivo csv y se fija un tamaño de 224x224x3, en 53 esta imagen es convertida a array, en 55 se normaliza dividiendo esta imagen ahora convertida en array entre 255 y por último en 57 se añade al array creado antes del ciclo for.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 64 de 104

```

1  import keras
2  import time
3  import numpy as np
4  import pandas as pd
5  import matplotlib.pyplot as plt
6
7  from keras.models import Sequential
8  from tensorflow.keras.applications.vgg16 import VGG16
9  from tensorflow.keras.applications.xception import Xception
10 from tensorflow.keras.applications.resnet import ResNet50
11 from tensorflow.keras.applications.inception_v3 import InceptionV3
12 from tensorflow.keras.applications.densenet import DenseNet121
13 from tensorflow.keras.applications.resnet import ResNet101
14 from tensorflow.keras.applications.efficientnet import EfficientNetB7
15 from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2
16 from keras.layers import Dense, InputLayer, Dropout, Flatten
17 from keras.layers import Conv2D, MaxPooling2D, GlobalMaxPooling2D
18 from keras.preprocessing import image
19 from tqdm import tqdm
20 from sklearn.model_selection import train_test_split
21
22 t = time.localtime()
23 current_time = time.strftime("%H_%M_%S", t)
24
25 # defining a function to save the weights of best model
26 from keras.callbacks import ModelCheckpoint
27
28 choice_of_archs = "VGG16"
29 epochs = 100
30
31 architectures = {
32     "VGG16": VGG16(weights='imagenet', include_top=False),
33     "ResNet50": ResNet50(weights='imagenet', include_top=False),
34     "InceptionV3": InceptionV3(weights='imagenet', include_top=False),
35     "Xception": Xception(weights='imagenet', include_top=False),
36     "DenseNet121": DenseNet121(weights='imagenet', include_top=False),
37     "ResNet101": ResNet101(weights='imagenet', include_top=False),
38     "EfficientNetB7": EfficientNetB7(weights='imagenet', include_top=False),
39     "MobileNetV2": MobileNetV2(weights='imagenet', include_top=False)
40 }
41
42 train = pd.read_csv('/content/drive/MyDrive/RobustFallDetection_Dataset_Copia/RGB/train_new2.csv')
43 train.head()
44
45 # creating an empty list
46 train_image = []
47
48 # for loop to read and store frames
49 for i in tqdm(range(train.shape[0])):
50     # loading the image and keeping the target size as (224,224,3)
51     img = image.load_img('/content/drive/MyDrive/RobustFallDetection_Dataset_Copia/train_2/'+train['image'][i], target_size=(224,224,3))
52     # converting it to array
53     img = image.img_to_array(img)
54     # normalizing the pixel value
55     img = img/255
56     # appending the image to the train_image list
57     train_image.append(img)
58

```

Figura 49. Lectura de csv y agregación de imágenes a la lista de entrenamiento

4.2. Convertir la lista de imágenes de entrenamiento a numpy array (línea 60). Separar las clases (línea 66). Crear el conjunto de entrenamiento y de validación (línea 69). Dumificar las variables para el conjunto de entrenamiento y validación (líneas 72 y 73) (ver Figura 50).

Crear el modelo base a partir del modelo pre entrenado de la arquitectura de red en uso (línea 76). Extraer características para las muestras de entrenamiento y para validación: la línea 79 hace

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 65 de 104

una predicción del modelo base y la línea 81 almacena el tamaño de la última capa de la arquitectura de red convolucional en uso. La línea 84 realiza la misma función de la línea 79 pero esta vez para la validación. Redimensionar las muestras de entrenamiento y de validación en una dimensión (líneas 88 y 89). Normalizar el valor de pixel: la línea 92 obtiene el valor máximo de la variable X_train que se usará para dividir X_train y X_test en las líneas 93 y 94 (características de entrenamiento y validación) (ver Figura 51).

```

59 # converting the list to numpy array
60 X = np.array(train_image)
61
62 # shape of the array
63 print(X.shape)
64
65 # separating the target
66 y = train['class']
67
68 # creating the training and validation set
69 X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.2, stratify = y)
70
71 # creating dummies of target variable for train and validation set
72 y_train = pd.get_dummies(y_train)
73 y_test = pd.get_dummies(y_test)

```

Figura 50. Entrenamiento parte #1

```

75 # creating the base model of pre-trained architecture model
76 base_model = architectures[choice_of_archs]
77
78 # extracting features for training frames
79 X_train = base_model.predict(X_train)
80 print(X_train.shape)
81 val_X_train = X_train.shape[1]*X_train.shape[2]*X_train.shape[3]
82
83 # extracting features for validation frames
84 X_test = base_model.predict(X_test)
85 print(X_test.shape)
86
87 # reshaping the training as well as validation frames in single dimension
88 X_train = X_train.reshape(X_train.shape[0], val_X_train)
89 X_test = X_test.reshape(X_test.shape[0], val_X_train)
90
91 # normalizing the pixel values
92 max = X_train.max()
93 X_train = X_train/max
94 X_test = X_test/max
95
96 # shape of images
97 print(X_train.shape)

```

Figura 51. Entrenamiento parte #2

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 66 de 104

4.3. Definir el modelo de arquitectura (línea 100) y se añaden las capas con las que se desea trabajar (líneas 101 – 109). Guardar el peso correspondiente de la arquitectura y la hora actual del proceso en un directorio establecido por el usuario (línea 114), la línea 112 concatena el directorio donde se almacenará el peso, el nombre de la arquitectura con que se realizó el entrenamiento, la hora actual y la extensión del archivo. Compilar y entrenar el modelo (líneas 116 y 118) (ver Figura 52).

```

99 #defining the model architecture
100 model = Sequential()
101 model.add(Dense(1024, activation='relu', input_shape=(val_X_train,)))
102 model.add(Dropout(0.5))
103 model.add(Dense(512, activation='relu'))
104 model.add(Dropout(0.5))
105 model.add(Dense(256, activation='relu'))
106 model.add(Dropout(0.5))
107 model.add(Dense(128, activation='relu'))
108 model.add(Dropout(0.5))
109 model.add(Dense(2, activation='softmax'))
110
111 #Save corresponding architecture weight + current time
112 filename_weight = '/content/drive/MyDrive/RobustFallDetection_Dataset_Copia/Weights/weight_'+choice_of_archs+'_'+current_time+'.hdf5'
113
114 mcp_save = ModelCheckpoint(filename_weight, save_best_only=True, monitor='val_loss', mode='min')
115 # compiling the model
116 model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy'])
117 # training the model
118 history = model.fit(X_train, y_train, epochs=epochs, validation_data=(X_test, y_test), callbacks=[mcp_save], batch_size=128)

```

Figura 52. Entrenamiento parte #3

5. Finalmente, se realiza la validación o testeo donde la red neuronal hará una comparación entre la realidad y su aprendizaje, es decir, a partir de un vídeo donde se tenga una acción, sea ADL o caída. Ser capaz de identificar dicha acción correctamente o, en su defecto, con el menor error posible. Esta fase de validación cuenta con los siguientes pasos:

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 67 de 104

1. Importar las librerías necesarias (líneas 1 – 26). Definir el archivo de pesos de la arquitectura a usar especificando su directorio y nombre (línea 28) y especificar el nombre de dicha arquitectura (línea 29) almacenada en el diccionario (líneas 32 – 41), se continúa con el uso de un *image shape* de 224x224x3 (línea 31) (ver Figura 53).

```

1  import numpy as np
2  import pandas as pd
3  import tensorflow as tf
4  import matplotlib.pyplot as plt
5  import cv2
6  import math
7  import os
8  import seaborn as sns
9
10 from keras.models import Sequential
11 from keras.layers import Dense, Dropout, Flatten
12 from keras.layers import Conv2D, MaxPooling2D
13 from keras.preprocessing import image
14 from tqdm import tqdm
15 from tensorflow.keras.applications.vgg16 import VGG16
16 from tensorflow.keras.applications.xception import Xception
17 from tensorflow.keras.applications.resnet import ResNet50
18 from tensorflow.keras.applications.inception_v3 import InceptionV3
19 from tensorflow.keras.applications.densenet import DenseNet121
20 from tensorflow.keras.applications.resnet import ResNet101
21 from tensorflow.keras.applications.efficientnet import EfficientNetB7
22 from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2
23 from glob import glob
24 from scipy import stats as s
25 # checking the accuracy of the predicted tags
26 from sklearn.metrics import accuracy_score, confusion_matrix, cohen_kappa_score, recall_score
27
28 path_to_weights = "/content/drive/MyDrive/RobustFallDetection_Dataset_Copia/Weights/weight_VGG16_23_30_09.hdf5"
29 choice_of_archs = "VGG16"
30
31 IMG_SHAPE = (224, 224, 3)
32 architectures = {
33     "VGG16": VGG16(input_shape = IMG_SHAPE, weights='imagenet', include_top=False),
34     "ResNet50": ResNet50(input_shape = IMG_SHAPE, weights='imagenet', include_top=False),
35     "InceptionV3": InceptionV3(input_shape = IMG_SHAPE, weights='imagenet', include_top=False),
36     "Xception": Xception(input_shape = IMG_SHAPE, weights='imagenet', include_top=False),
37     "DenseNet121": DenseNet121(input_shape = IMG_SHAPE, weights='imagenet', include_top=False),
38     "ResNet101": ResNet101(input_shape = IMG_SHAPE, weights='imagenet', include_top=False),
39     "EfficientNetB7": EfficientNetB7(input_shape = IMG_SHAPE, weights='imagenet', include_top=False),
40     "MobileNetV2": MobileNetV2(input_shape = IMG_SHAPE, weights='imagenet', include_top=False)
41 }

```

Figura 53. Carga de pesos y definición de arquitectura de red

- Al momento de definir el modelo base, es preciso especificar la última capa de la arquitectura a trabajar y, de esta manera, realizar el producto de sus pesos para obtener el *inputshape* requerido. Para obtener el nombre de la capa requerida es necesario

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 68 de 104

valerse de la función *summary* y usarla en el modelo base de la arquitectura de red (ver Figura 54).

```
base_model.summary()
block4_conv2 (Conv2D)      (None, 28, 28, 512)      2359808
block4_conv3 (Conv2D)      (None, 28, 28, 512)      2359808
block4_pool (MaxPooling2D) (None, 14, 14, 512)      0
block5_conv1 (Conv2D)      (None, 14, 14, 512)      2359808
block5_conv2 (Conv2D)      (None, 14, 14, 512)      2359808
block5_conv3 (Conv2D)      (None, 14, 14, 512)      2359808
block5_pool (MaxPooling2D) (None, 7, 7, 512)       0
=====
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
```

Figura 54. Resumen de la arquitectura de red neuronal especificada (Figura de VGG16)

3. Una vez obtenido el nombre de la última capa de la arquitectura de red neuronal convolucional, se le especifica al modelo base tanto la arquitectura con la que se va a trabajar (línea 43) como el nombre de la última capa obtenido previamente (línea 44) y se realiza el producto de los pesos de la capa (línea 45), seguidamente, se define el modelo de arquitectura (línea 48) y las capas a trabajar de la misma forma que se hizo en el entrenamiento (líneas 49 – 57), pasando como parámetro el inputshape hallado en el paso anterior (línea 49). Cabe resaltar que tanto para la validación como para el entrenamiento se usan activaciones ReLU y Softmax (ver Figura 55).

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 69 de 104

```

43 base_model = architectures[choice_of_archs]
44 t = base_model.get_layer('block5_pool').output.get_shape()
45 InputShape = t[1]*t[2]*t[3]
46
47 #defining the model architecture
48 model = Sequential()
49 model.add(Dense(1024, activation='relu', input_shape=(InputShape,)))
50 model.add(Dropout(0.5))
51 model.add(Dense(512, activation='relu'))
52 model.add(Dropout(0.5))
53 model.add(Dense(256, activation='relu'))
54 model.add(Dropout(0.5))
55 model.add(Dense(128, activation='relu'))
56 model.add(Dropout(0.5))
57 model.add(Dense(2, activation='softmax'))

```

Figura 55. Especificación de nombre de capa de la arquitectura de red, realización del producto de sus pesos y definición de modelo de arquitectura

4. Se cargan los pesos pre-entrenados de la arquitectura en la fase anterior de entrenamiento (línea 60) y se compila el modelo (línea 62). A continuación, se obtiene la lista de entrenamiento, se lee la misma y se separan los nombres de los vídeos en ella (líneas 65 - 67), se crea el dataframe donde se almacenarán los nombres de los vídeos (líneas 70 – 73) y se crean las etiquetas de ADL y FALL extrayéndolas del archivo csv que contiene cada muestra de los vídeos con su respectiva categoría y se dumifican (líneas 77 – 79). Se crean dos listas para almacenar las etiquetas de predicción y actuales (líneas 82 y 83) (ver Figura 56).

```

59 # loading the trained weights
60 model.load_weights(path_to_weights)
61 # compiling the model
62 model.compile(loss='categorical_crossentropy',optimizer='Adam',metrics=['accuracy'])
63
64 # getting the test list
65 f = open("/content/drive/MyDrive/RobustFallDetection_Dataset_Copia/RGBTrainTestList/testlist01.txt", "r")
66 temp = f.read()
67 videos = temp.split('\n')
68
69 # creating the dataframe
70 test = pd.DataFrame()
71 test['video_name'] = videos
72 test = test[:-1]
73 test_videos = test['video_name']
74 #print(test.head())
75
76 # creating the tags
77 train = pd.read_csv('/content/drive/MyDrive/RobustFallDetection_Dataset_Copia/RGB/train_new2.csv')
78 y = train['class']
79 y = pd.get_dummies(y)
80
81 # creating two lists to store predicted and actual tags
82 predict = []
83 actual = []

```

Figura 56. Carga de pesos, obtención de lista de validación, creación de dataframe, etiquetas y listas de almacenamiento

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 70 de 104

5. Se extraen *frames* de cada vídeo de validación, definiendo un bucle (línea 86) y especificando el directorio donde se encuentran mediante una capturadora (línea 91). Aquí se ve con más claridad por qué se debía ajustar la cantidad de elementos del archivo csv, ya que es en este punto donde los vídeos leídos deben coincidir con la cantidad de *frames*, haciendo un emparejamiento 1 a 1. El *frame rate* establecido es de 5 por cada vídeo (línea 92). Se elimina cualquier archivo contenido en el directorio temp (líneas 95 – 97) y se almacenan los *frames* extraídos de cada vídeo únicamente en esta fase del algoritmo en un directorio definido por el usuario mediante la creación de un ciclo while que estará activo, extrayendo *frames* y almacenándolos mientras la capturadora de videos se encuentre abierta (líneas 98 – 106), una vez el vídeo no permita más extracción de *frames*, la capturadora se liberará (línea 107) terminando un ciclo para un vídeo y continuando con el siguiente; luego, se leen todos los *frames* almacenados en este directorio (línea 110), se procesan adentro de otro bucle que se encargará de cargar cada *frame* (línea 114), convertirlo en array (línea 115), normalizándolo al dividirlo entre 255 (línea 116) y añadiéndolo a una lista de imágenes de predicción (línea 117) (ver Figura 57).

```

85 # for loop to extract frames from each test video
86 for i in tqdm(range(test_videos.shape[0])):
87     count = 0
88     videoFile = test_videos[i]
89
90     # capturing the video from the given path
91     cap = cv2.VideoCapture('/content/drive/MyDrive/RobustFallDetection_Dataset_Copia/RGB/'+train['class'][i]+'/' + videoFile.split(' ')[0].split('/')[1])
92     frameRate = cap.get(5) #frame rate
93     x=1
94     # removing all other files from the temp folder
95     files = glob('/content/drive/MyDrive/RobustFallDetection_Dataset_Copia/temp/*')
96     for f in files:
97         os.remove(f)
98     while(cap.isOpened()):
99         frameId = cap.get(1) #current frame number
100         ret, frame = cap.read()
101         if (ret != True):
102             break
103         if (frameId % math.floor(frameRate) == 0):
104             # storing the frames of this particular video in temp folder
105             filename = '/content/drive/MyDrive/RobustFallDetection_Dataset_Copia/temp/' + "_frame%d.jpg" % count; count+=1
106             cv2.imwrite(filename, frame)
107         cap.release()
108
109     # reading all the frames from temp folder
110     images = glob("/content/drive/MyDrive/RobustFallDetection_Dataset_Copia/temp/*.jpg")
111
112     prediction_images = []
113     for i in range(len(images)):
114         img = image.load_img(images[i], target_size=IMG_SHAPE)
115         img = image.img_to_array(img)
116         img = img/255
117         prediction_images.append(img)

```

Figura 57. Bucle de procesamiento de muestras de vídeo

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 71 de 104

6. Se convierten todos los *frames* de validación en numpy array (línea 121), se extraen características usando el modelo pre entrenado (línea 123) y se convierten a un array unidimensional (línea 125). Se predicen las etiquetas para cada array (línea 127), luego se agrega el modo de predicción a la lista de predicciones para asignar la etiqueta correspondiente al vídeo (línea 130) y se añade la etiqueta actual al vídeo (línea 132) (ver Figura 58). De esta forma, se logra evidenciar qué tan exacta y precisa es la red neuronal con el RFDD.

```

119     # converting all the frames for a test video into numpy array
120     #print(len(prediction_images))
121     prediction_images = np.array(prediction_images)
122     # extracting features using pre-trained model
123     prediction_images = base_model.predict(prediction_images)
124     # converting features in one dimensional array
125     prediction_images = prediction_images.reshape(prediction_images.shape[0], InputShape)
126     # predicting tags for each array
127     prediction = model.predict_classes(prediction_images)
128     #print(prediction)
129     # appending the mode of predictions in predict list to assign the tag to the video
130     predict.append(y.columns.values[s.mode(prediction)[0][0]])
131     # appending the actual tag of the video
132     actual.append(videoFile.split('/')[1].split('_')[1])

```

Figura 58. Procesamiento de frames y predicción de la data

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 72 de 104

CAPÍTULO 5 - RESULTADOS

5. RESULTADOS

5.1. ENTRENAMIENTO

Los resultados obtenidos en el entrenamiento y la validación se mostrarán en tablas y gráficas para su mejor comprensión. Todas las arquitecturas fueron entrenadas durante 100 épocas, los resultados que se muestran en la Tabla 2 pertenecen a las métricas de validación obtenidas durante el proceso de entrenamiento de cada arquitectura con el conjunto de datos RFDD.

Tabla 2. Resultados de entrenamiento con diferentes arquitecturas de red neuronal

ARCH	LOSS	ACCURACY	VAL_LOSS	VAL_ACCURACY
VGG16	0.0136	0.9952	0.1788	0.9714
ResNet50	0.4739	0.8237	0.4770	0.8208
InceptionV3	0.0015	0.9992	0.2498	0.9714
Xception	0.0014	1.0000	0.3242	0.9662
DenseNet121	0.0040	0.9991	0.2382	0.9740
ResNet101	0.3757	0.8186	0.2746	0.8208
EfficientNetB7	0.4649	0.8250	0.4702	0.8208
MobileNetV2	0.00093	1.0000	0.1876	0.9766

La tabla 2, consolida las métricas de validación relacionadas con el entrenamiento de 8 arquitecturas de redes neuronales convolucionales, las métricas que se tuvieron en cuenta fueron la pérdida expresada por la entropía categórica cruzada y la precisión tanto para la etapa de entrenamiento y validación. La arquitectura que presentó las mejores métricas fue MobileNetV2, teniendo los mejores resultados en 3 de las 4 categorías a evaluar, y la arquitectura que tuvo menor desempeño fue ResNet50 al obtener los resultados más bajos en 3 de las 4 categorías a evaluar.

A continuación, se presentan las figuras que muestran la evolución de la función de pérdida y precisión en las etapas de entrenamiento y validación para cada arquitectura de red neuronal (Figuras 59 – 66):

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 73 de 104

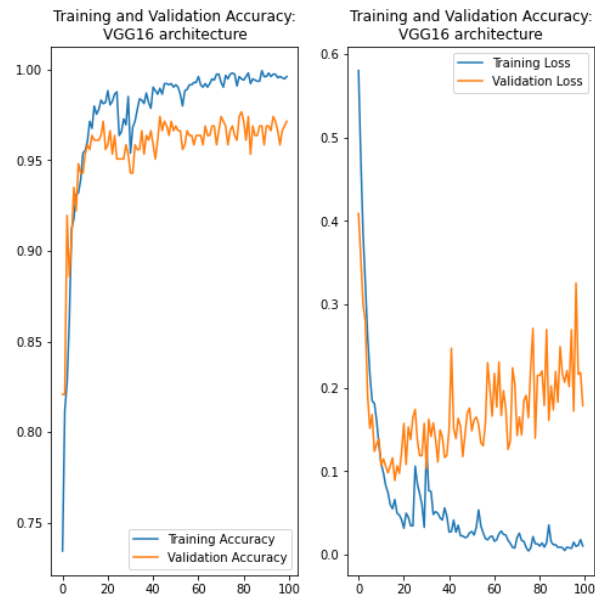


Figura 59. Métricas de evaluación: pérdida y precisión para la arquitectura VGG16 resolviendo el conjunto de datos RFDD.

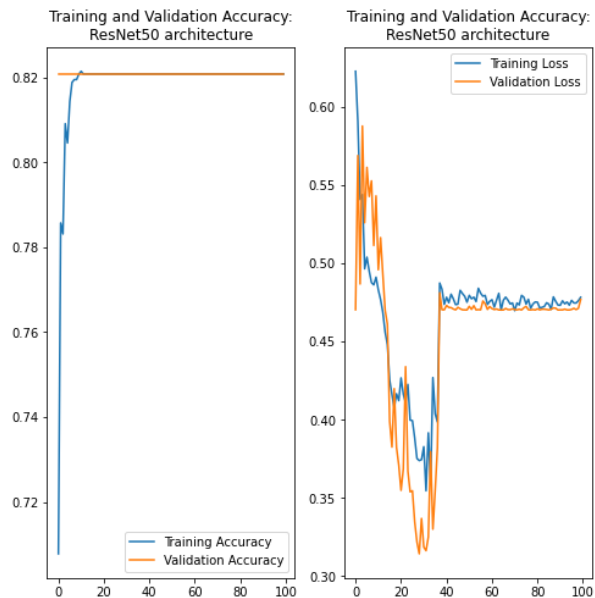


Figura 60. Métricas de evaluación: pérdida y precisión para la arquitectura ResNet50 resolviendo el conjunto de datos RFDD.

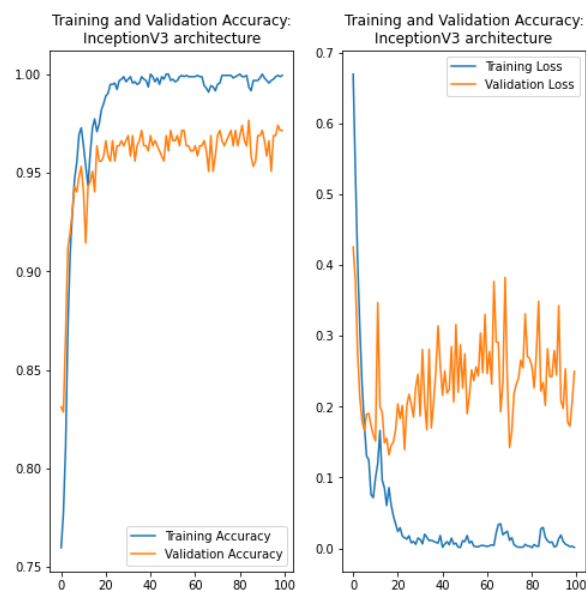


Figura 61. Métricas de evaluación: pérdida y precisión para la arquitectura InceptionV3 resolviendo el conjunto de datos RFDD.

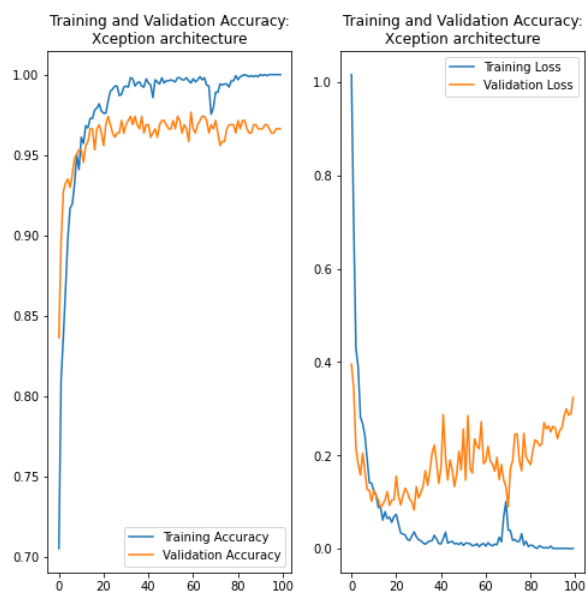


Figura 62. Métricas de evaluación: pérdida y precisión para la arquitectura Xception resolviendo el conjunto de datos RFDD.

DOCUMENTO FINAL

PROYECTO DE GRADO

EDO – 02

Versión 1

Vigencia:
30/06/2021

Página 75 de 104

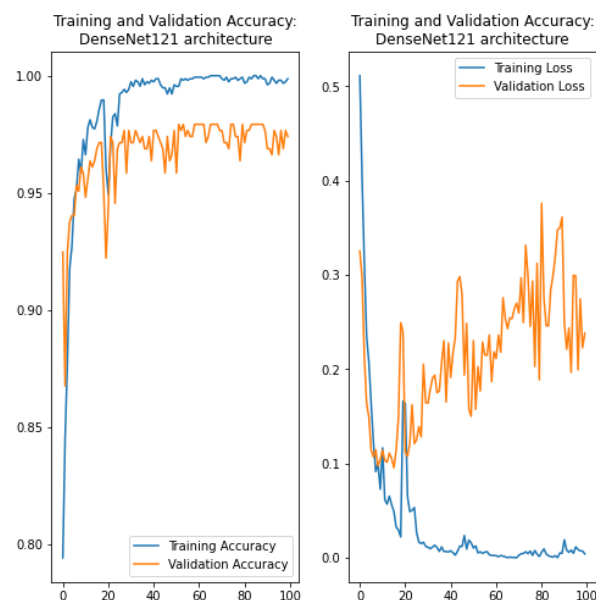


Figura 63. Métricas de evaluación: pérdida y precisión para la arquitectura DenseNet121 resolviendo el conjunto de datos RFDD.

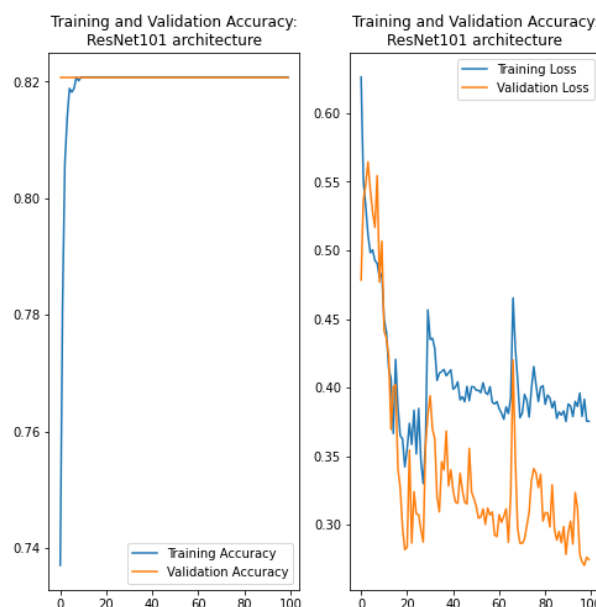


Figura 64. Métricas de evaluación: pérdida y precisión para la arquitectura ResNet101 resolviendo el conjunto de datos RFDD.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 76 de 104

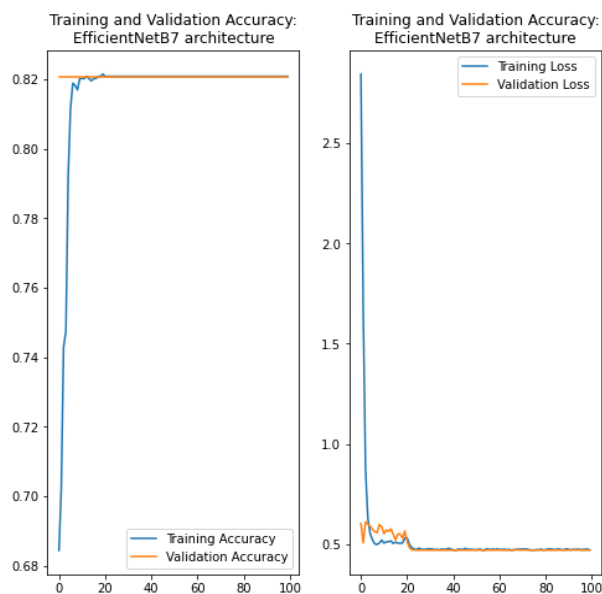


Figura 65. Métricas de evaluación: pérdida y precisión para la arquitectura EfficientNetB7 resolviendo el conjunto de datos RFDD.

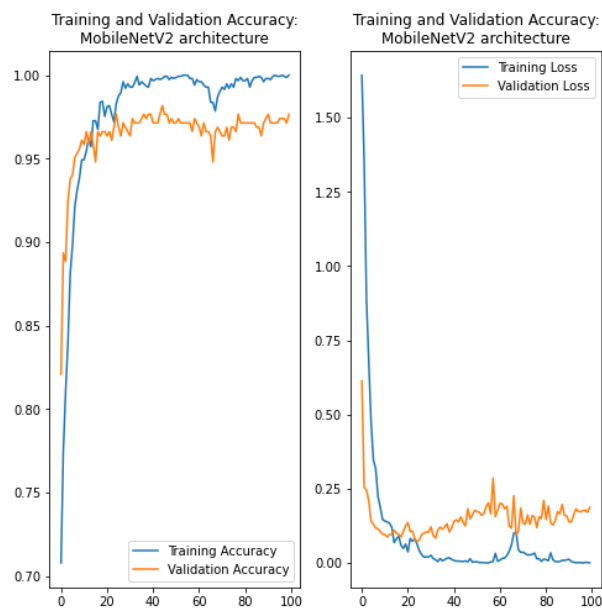


Figura 66. Métricas de evaluación: pérdida y precisión para la arquitectura MobileNetV2 resolviendo el conjunto de datos RFDD.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 77 de 104

5.2. VALIDACIÓN

Tabla 3. Resultados de validación con diferentes arquitecturas de red neuronal

ARCH	TIEMPO DE VALIDACIÓN (HH:MM:SS)	OVERALL ACCURACY (%)	COEFICIENTE KAPPA
VGG16	1:46:11	91.7922	0.7235
ResNet50	1:05:13	82.0779	0.0
InceptionV3	0:44:37	91.4805	0.7205
Xception	1:04:51	91.2207	0.7184
DenseNet121	1:03:51	91.3766	0.7226
ResNet101	1:49:04	82.0779	0.0
EfficientNetB7	2:08:33	82.0779	0.0
MobileNetV2	0:46:57	91.7922	0.7313

La tabla 3, consolida las métricas de validación relacionadas con la validación de 8 arquitecturas de redes neuronales convolucionales, las métricas que se tuvieron en cuenta fueron el tiempo de validación, la precisión total y el coeficiente Kappa. La arquitectura que presentó las mejores métricas de fue nuevamente MobileNetV2 al destacar en 2 de las 3 categorías y la arquitectura que menor desempeño tuvo fue EfficientNetB7 al quedar por debajo de las demás arquitecturas en las 3 categorías a evaluar.

A continuación, se presentan las figuras que muestran las matrices de confusión para cada arquitectura de red neuronal (Figuras 67 – 74):

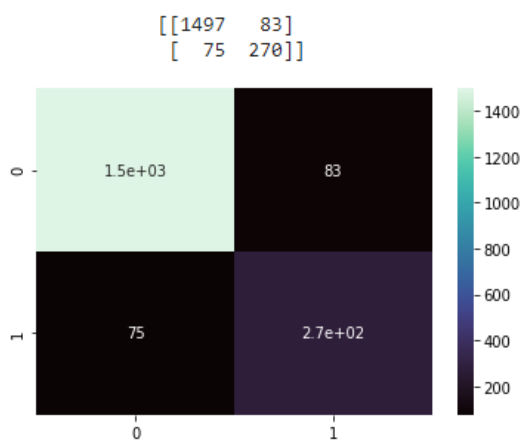


Figura 67. Matriz de confusión VGG16

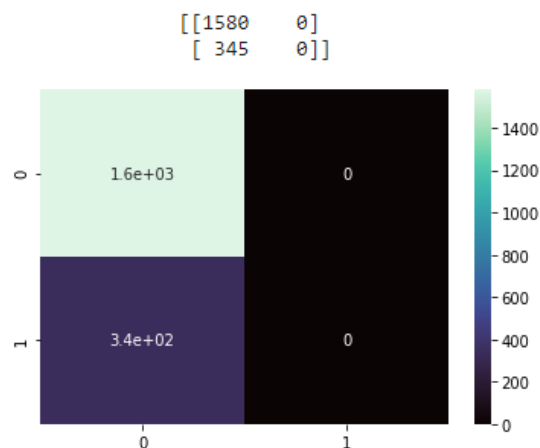


Figura 68. Matriz de confusión ResNet50

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 78 de 104

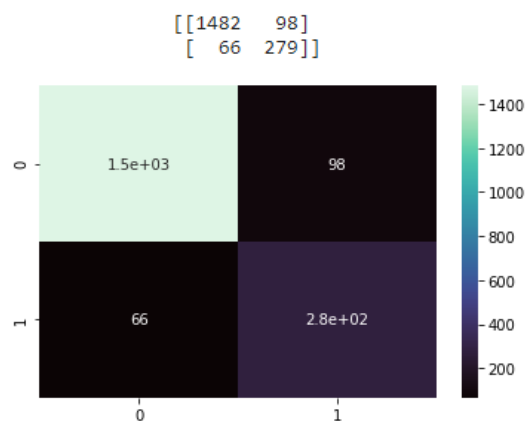


Figura 69. Matriz de confusión InceptionV3

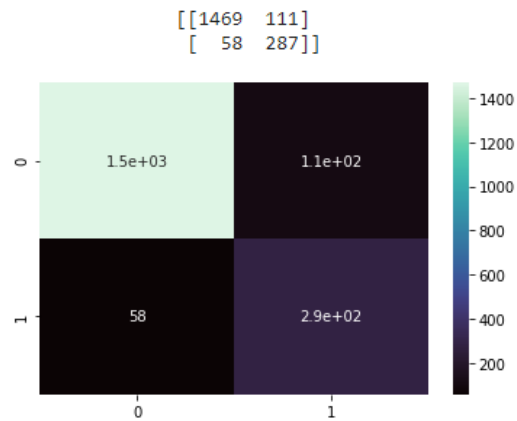


Figura 70. Matriz de confusión Xception

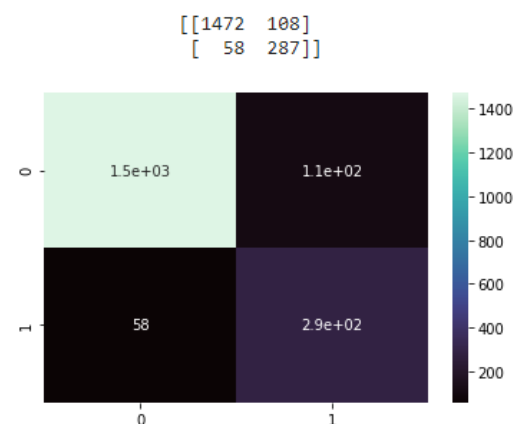


Figura 71. Matriz de confusión DenseNet121

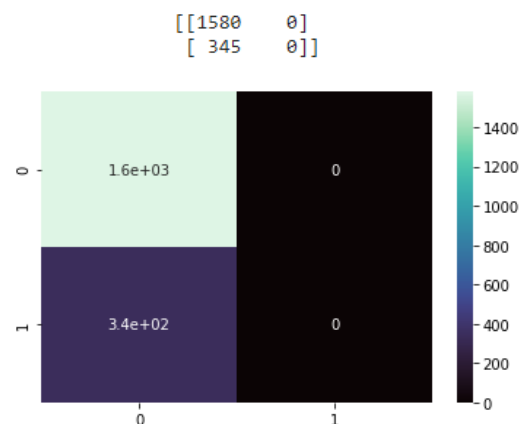


Figura 72. Matriz de confusión ResNet101

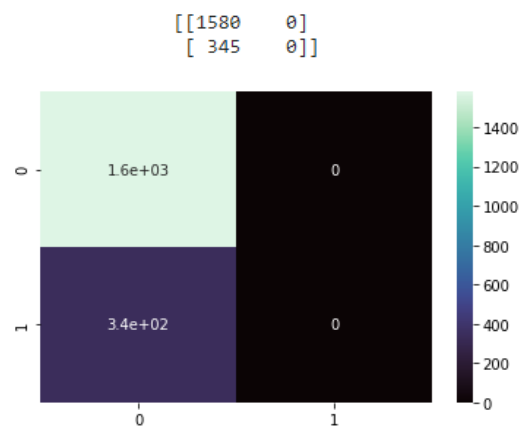


Figura 73. Matriz de confusión EfficientNetB7

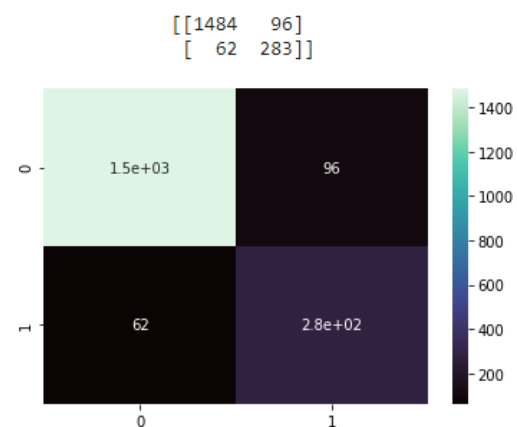


Figura 74. Matriz de confusión MobileNetV2

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 79 de 104

CAPÍTULO 6

CONCLUSIONES

En este proyecto de investigación se establecen las condiciones requeridas y de calidad relacionadas con los videos de acciones de caída (FALL) o acciones correspondientes al diario vivir (ADL) para poder crear conjuntos de datos robustos que aporten significativamente a proyectos relacionados con la detección de caídas de personas en diferentes tipos de ambientes. Se definió unas características importantes que se deben cumplir para evitar que los entrenamientos de las redes neuronales profundas sufran un estancamiento en su proceso de optimización de la función de pérdida del algoritmo. En este trabajo se construye el primer conjunto de datos robusto relacionado con caídas de personas, el conjunto de datos es nombrado **RFDD (Robust Fall Detection Dataset)** y se caracteriza por tener una diversidad de actores, ambientes y tipo de información espectral en las imágenes que lo componen: RGB e Infrarrojo.

El conjunto de datos **RFDD** fue construido basándose en diferentes datasets validados en el estado del arte a través de la revisión por pares en revistas indexadas y con licencias permisivas en su manipulación y descarga. La metodología adoptada en la investigación permitió que se generaran acciones sobre los distintos tipos de datos con el fin de optimizar procesos de cómputo: entrenamiento de redes neuronales convolucionales; procesos de almacenamiento: eliminar escenas y acciones que no aportan al proceso de clasificación de caídas de personas y favoreciendo el almacenamiento en disco a través de la creación de data inteligente antes que grandes volúmenes de información con ruido en sus muestras.

Al usar diferentes arquitecturas de redes neuronales, se puso a prueba la **eficiencia** del **RFDD** mediante un algoritmo de clasificación, arrojando resultados considerablemente positivos en el entrenamiento, siendo 82,08 % de precisión el valor más bajo, y 97,66 % de precisión el más alto a cargo de la arquitectura MobileNetV2, demostrando de esta forma que el **RFDD** puede ser

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 80 de 104

entrenado de manera correcta y con excelentes resultados. Por otra parte, en la etapa de validación, las dos arquitecturas que destacaron por encima del resto fueron VGG16 y MobileNetV2, con una precisión del 91,79 % al momento de clasificar en ADL o FALL, las acciones dentro de un vídeo, con 1497 verdaderos positivos, 83 falsos positivos, 75 falsos negativos, y 270 verdaderos negativos usando VGG16 y 1484 verdaderos positivos, 96 falsos positivos, 62 falsos negativos y 283 verdaderos negativos con MobileNetV2, ambas de un total de 1925 vídeos comprendidos entre acciones de ADL y caídas.

Individualmente, cada dataset usado para la creación del **RFDD** arroja resultados positivos al momento del entrenamiento y validación. Lastimosamente, no todos los investigadores tendrán preferencia por usar alguno de estos conjuntos de datos debido a las falencias que puedan tener como el contener más de una acción en una muestra. Con la construcción del **RFDD** se facilita al investigador un conjunto de gran tamaño de archivos de vídeo con información filtrada y preprocesada para eliminar el ruido que afecta los procesos de clasificación, el investigador podrá decidir qué tipo de data usar según sus necesidades (RGB o IR), a la vez que ahorrará tiempo y costos en producción de un dataset creado desde cero.

No se puede afirmar con plena seguridad que las arquitecturas de red neuronal convolucional que tuvieron menor desempeño con el **RFDD** no sean confiables o que no puedan resolver el dataset, un proceso de optimización de los hiperparámetros debería ser realizado como trabajo a futuro para dar una mejor visión sobre cuál sería la mejor arquitectura, sin embargo, en este proyecto se planteó un escenario de comparación justa teniendo siempre controlada la semilla de inicialización del algoritmo, la tasa de aprendizaje y la cantidad de épocas que entrenará cada arquitectura. Queda a criterio del investigador la selección de la arquitectura según su tiempo y recursos computacionales disponibles.

La posibilidad de crear una segunda versión del **RFDD** queda abierta, la adición de nuevos casos de caídas y/o actividades del diario vivir deberían cumplir con las condiciones establecidas que **RFDD** presenta y que fueron nombradas en la metodología del proyecto. **RFDD** quedará

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 81 de 104

disponible y bajo licencia Creative Commons para alentar a la comunidad científica e interesada en el campo de la detección de caídas a robustecer el dataset construido para seguir aportando en temas relacionados con la seguridad del paciente.

Para futuras investigaciones, se plantea la idea de realizar una comparativa entre diferentes datasets de detección de caídas y el RFDD usando las mismas métricas de este proyecto o las que el grupo de investigación precise. Lo ideal sería utilizar una gran diversidad de datasets con diferentes ambientes y tipos de muestras ya que, de esta forma, el RFDD estaría mejor puesto a prueba.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 82 de 104

REFERENCIAS

- [1] WHO, “Falls,” 2018. <https://www.who.int/news-room/fact-sheets/detail/falls> (accessed Nov. 12, 2020).
- [2] Banco Mundial, “Población de 65 años de edad y más,” 2020.
<https://datos.bancomundial.org/indicador/SP.POP.65UP.TO.ZS?end=2019&start=2017>
(accessed Nov. 12, 2020).
- [3] WHO, “Datos interesantes acerca del envejecimiento,” *WHO*, 2015, Accessed: Nov. 12, 2020. [Online]. Available: <http://www.who.int/ageing/about/facts/es/>.
- [4] E. Casilari, R. Lora-rivera, and F. García-lagos, “A study on the application of convolutional neural networks to fall detection evaluated with multiple public datasets,” *Sensors (Switzerland)*, vol. 20, no. 5, Mar. 2020, doi: 10.3390/s20051466.
- [5] Revista Semana, “El riesgo de morir por una caída es mayor de lo que se cree,” 2018.
<https://www.semana.com/vida-moderna/articulo/panorama-de-muertes-por-caidas-en-colombia/571281/> (accessed Nov. 12, 2020).
- [6] S. Gasparrini *et al.*, “Proposal and experimental evaluation of fall detection solution based on wearable and depth data fusion,” in *Advances in Intelligent Systems and Computing*, 2016, vol. 399, pp. 99–108, doi: 10.1007/978-3-319-25733-4_11.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 83 de 104

- [7] X. Kong, Z. Meng, N. Nojiri, Y. Iwahori, L. Meng, and H. Tomiyama, “A HOG-SVM Based Fall Detection IoT System for Elderly Persons Using Deep Sensor,” in *Procedia Computer Science*, Jan. 2019, vol. 147, pp. 276–282, doi: 10.1016/j.procs.2019.01.264.
- [8] L. Chen, X. Kong, H. Tomiyama, and L. Meng, “Multiple States Fall Detection System for Senior Citizens,” in *International Conference on Advanced Mechatronic Systems, ICAMechS*, Aug. 2019, vol. 2019-Augus, pp. 169–174, doi: 10.1109/ICAMechS.2019.8861658.
- [9] L. Yao, W. Yang, and W. Huang, “An improved feature-based method for fall detection,” *Teh. Vjesn.*, vol. 26, no. 5, pp. 1363–1368, Oct. 2019, doi: 10.17559/TV-20190411015902.
- [10] A. Takebayashi *et al.*, “Fall Detection of Elderly Persons by Action Recognition Using Data Augmentation and State Transition Diagram,” in *Studies in Computational Intelligence*, 2020, vol. 847, pp. 95–109, doi: 10.1007/978-3-030-25217-5_8.
- [11] P. Bauer, J. B. Kramer, B. Rush, and L. Sabalka, “Modeling bed exit likelihood in a camera-based automated video monitoring application,” in *IEEE International Conference on Electro Information Technology*, Sep. 2017, pp. 56–61, doi: 10.1109/EIT.2017.8053330.
- [12] D. P. Betancourt, C. J. R. Restrepo, C. I. C. Valencia, R. A. E. López, C. A. M. Cubillos, and L. F. C. Serna, “DIEGO PALACIO BETANCOURT Ministro de la Protección

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 84 de 104

Social,” 2010.

- [13] S. S. Khan and J. Hoey, “Review of fall detection techniques: A data availability perspective,” *Medical Engineering and Physics*, vol. 39. Elsevier Ltd, pp. 12–22, Jan. 01, 2017, doi: 10.1016/j.medengphy.2016.10.014.
- [14] G. Vavoulas, M. Padiaditis, E. G. Spanakis, and M. Tsiknakis, “The MobiFall dataset: An initial evaluation of fall detection algorithms using smartphones,” 2013, doi: 10.1109/BIBE.2013.6701629.
- [15] B. Kwolek and M. Kepski, “Human fall detection on embedded platform using depth maps and wireless accelerometer,” *Comput. Methods Programs Biomed.*, vol. 117, no. 3, pp. 489–501, Dec. 2014, doi: 10.1016/j.cmpb.2014.09.005.
- [16] T. Vilarinho *et al.*, “A combined smartphone and smartwatch fall detection system,” in *Proceedings - 15th IEEE International Conference on Computer and Information Technology, CIT 2015, 14th IEEE International Conference on Ubiquitous Computing and Communications, IUCC 2015, 13th IEEE International Conference on Dependable, Autonomic and Se*, 2015, pp. 1443–1448, doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.216.
- [17] A. Sucerquia, J. D. López, and J. F. Vargas-Bonilla, “SisFall: A fall and movement dataset,” *Sensors (Switzerland)*, vol. 17, no. 1, Jan. 2017, doi: 10.3390/s17010198.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 85 de 104

- [18] K. Adhikari, H. Bouchachia, and H. Nait-Charif, “Activity recognition for indoor fall detection using convolutional neural network,” in *Proceedings of the 15th IAPR International Conference on Machine Vision Applications, MVA 2017*, 2017, pp. 81–84, doi: 10.23919/MVA.2017.7986795.
- [19] M. Ahmed, N. Mehmood, A. Nadeem, A. Mehmood, and K. Rizwan, “Fall detection system for the elderly based on the classification of shimmer sensor prototype data,” *Healthc. Inform. Res.*, vol. 23, no. 3, pp. 147–158, 2017, doi: 10.4258/hir.2017.23.3.147.
- [20] R. Rajagopalan, I. Litvan, and T. P. Jung, “Fall prediction and prevention systems: Recent trends, challenges, and future research directions,” *Sensors (Switzerland)*, vol. 17, no. 11. MDPI AG, Nov. 01, 2017, doi: 10.3390/s17112509.
- [21] M. Ricciuti, S. Spinsante, and E. Gambi, “Accurate fall detection in a top view privacy preserving configuration,” *Sensors (Switzerland)*, vol. 18, no. 6, Jun. 2018, doi: 10.3390/s18061754.
- [22] L. Martínez-Villaseñor, H. Ponce, J. Brieva, E. Moya-Albor, J. Núñez-Martínez, and C. Peñafort-Asturiano, “Up-fall detection dataset: A multimodal approach,” *Sensors (Switzerland)*, vol. 19, no. 9, May 2019, doi: 10.3390/s19091988.
- [23] X. Kong, L. Chen, Z. Wang, Y. Chen, L. Meng, and H. Tomiyama, “Robust self-adaptation fall-detection system based on camera height,” *Sensors (Switzerland)*, vol. 19, no. 17, Sep. 2019, doi: 10.3390/s19173768.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 86 de 104

- [24] A. Bezet, “LibGuides: Research Process: Datasets,” Accessed: Nov. 12, 2020. [Online]. Available: <https://ncu.libguides.com/researchprocess/datasets>.
- [25] K. Turgutlu, “Structured Deep Learning. Fast | by Kerem Turgutlu | Towards Data Science,” *Towards Data Science*, Nov. 24, 2017. <https://towardsdatascience.com/structured-deep-learning-b8ca4138b848> (accessed Jun. 10, 2021).
- [26] H. Tyagi, “Types of Structured Data Every Data Science Enthusiast should Know | by Harshit Tyagi | Towards Data Science,” *Towards Data Science*, Nov. 14, 2020. <https://towardsdatascience.com/types-of-structured-data-every-data-science-enthusiast-should-know-a656b95afbe2> (accessed Jun. 10, 2021).
- [27] T. Y. Lin *et al.*, “Microsoft COCO: Common objects in context,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, May 2014, vol. 8693 LNCS, no. PART 5, pp. 740–755, doi: 10.1007/978-3-319-10602-1_48.
- [28] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, “ImageNet: A large-scale hierarchical image database,” Mar. 2010, pp. 248–255, doi: 10.1109/cvpr.2009.5206848.
- [29] S. A. Velastin and D. A. Gómez-Lira, “People detection and pose classification inside a moving train using computer vision,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*,

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 87 de 104

Nov. 2017, vol. 10645 LNCS, pp. 319–330, doi: 10.1007/978-3-319-70010-6_30.

- [30] O. Ojetola, E. Gaura, and J. Brusey, “Data set for fall events and daily activities from inertial sensors,” in *Proceedings of the 6th ACM Multimedia Systems Conference, MMSys 2015*, Mar. 2015, pp. 243–248, doi: 10.1145/2713168.2713198.
- [31] K. Adhikari, H. Bouchachia, and H. Nait-Charif, “Activity recognition for indoor fall detection using convolutional neural network,” in *Proceedings of the 15th IAPR International Conference on Machine Vision Applications, MVA 2017*, Jul. 2017, pp. 81–84, doi: 10.23919/MVA.2017.7986795.
- [32] A. Sucerquia, J. D. López, and J. F. Vargas-Bonilla, “SisFall: A fall and movement dataset,” *Sensors (Switzerland)*, vol. 17, no. 1, Jan. 2017, doi: 10.3390/s17010198.
- [33] Z. Zhang, C. Conly, and V. Athitsos, “Evaluating depth-based computer vision methods for fall detection under occlusions,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8888, pp. 196–207, doi: 10.1007/978-3-319-14364-4_19.
- [34] E. Casilari, J. A. Santoyo-Ramón, and J. M. Cano-García, “Analysis of a smartphone-based architecture with multiple mobility sensors for fall detection,” *PLoS One*, vol. 11, no. 12, p. e0168069, Dec. 2016, doi: 10.1371/journal.pone.0168069.
- [35] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, Nature

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 88 de 104

Publishing Group, pp. 436–444, May 27, 2015, doi: 10.1038/nature14539.

- [36] I. Hadji and R. P. Wildes, “What do we understand about convolutional networks?,” *arXiv*. Mar. 23, 2018, Accessed: Nov. 12, 2020. [Online]. Available: <http://arxiv.org/abs/1803.08834>.
- [37] K. Simonyan and A. Zisserman, “VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION,” *arXiv*, vol. 75, no. 6, pp. 398–406, 2018.
- [38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017, doi: 10.1145/3065386.
- [39] M. Gurucharan, “Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network,” *upGrad blog*, Dec. 07, 2020. <https://www.upgrad.com/blog/basic-cnn-architecture/> (accessed Jun. 10, 2021).
- [40] M. ul Hassan, “VGG16 - Convolutional Network for Classification and Detection,” *neurohive*, 2018. <https://neurohive.io/en/popular-networks/vgg16/> (accessed Jun. 07, 2021).
- [41] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” Sep. 2015, Accessed: Jun. 07, 2021. [Online]. Available: <http://www.robots.ox.ac.uk/>.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 89 de 104

- [42] A. Parashar, “Vgg 16 Architecture, Implementation and Practical Use | by Abhay Parashar | The Pythoneers | Medium,” *Medium*, 2020. <https://medium.com/pythoneers/vgg-16-architecture-implementation-and-practical-use-e0fef1d14557> (accessed Jun. 07, 2021).
- [43] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Dec. 2016, vol. 2016-Decem, pp. 770–778, doi: 10.1109/CVPR.2016.90.
- [44] H. Mujtaba, “What is Resnet or Residual Network | How Resnet Helps?,” 2020. <https://www.mygreatlearning.com/blog/resnet/> (accessed Jun. 08, 2021).
- [45] G. Shi, “Implementing a ResNet model from scratch. - Towards Data Science,” *Towards Data Science*, 2019. <https://towardsdatascience.com/implementing-a-resnet-model-from-scratch-971be7193718> (accessed Jun. 08, 2021).
- [46] A. Milton-Barker, “Inception V3 Deep Convolutional Architecture For Classifying Acute....,” *Intel*, 2019. <https://software.intel.com/content/www/us/en/develop/articles/inception-v3-deep-convolutional-architecture-for-classifying-acute-myeloidlymphoblastic.html> (accessed Jun. 08, 2021).
- [47] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Dec. 2016, vol. 2016-Decem,

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 90 de 104

pp. 2818–2826, doi: 10.1109/CVPR.2016.308.

- [48] B. Raj, “A Simple Guide to the Versions of the Inception Network | by Bharath Raj | Towards Data Science.” 2018, Accessed: Jun. 08, 2021. [Online]. Available: <https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202>.
- [49] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, Nov. 2017, vol. 2017-Janua, pp. 1800–1807, doi: 10.1109/CVPR.2017.195.
- [50] A. Sarkar, “Xception: Implementing from scratch using Tensorflow | by Arjun Sarkar | Towards Data Science,” *Towards Data Science*, 2020.
<https://towardsdatascience.com/xception-from-scratch-using-tensorflow-even-better-than-inception-940fb231ced9> (accessed Jun. 08, 2021).
- [51] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, Nov. 2017, vol. 2017-Janua, pp. 2261–2269, doi: 10.1109/CVPR.2017.243.
- [52] S.-H. Tsang, “Review: DenseNet — Dense Convolutional Network (Image Classification),” <https://towardsdatascience.com/>, Nov. 25, 2018.
<https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803>

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 91 de 104

(accessed Jun. 08, 2021).

- [53] M. Tan and Q. V. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” in *36th International Conference on Machine Learning, ICML 2019*, May 2019, vol. 2019-June, pp. 10691–10700, Accessed: Jun. 08, 2021. [Online]. Available: <http://arxiv.org/abs/1905.11946>.
- [54] A. Borad, “Image Classification with EfficientNet: Better performance with computational efficiency,” *Medium*, 2019. <https://medium.com/analytics-vidhya/image-classification-with-efficientnet-better-performance-with-computational-efficiency-f480fdb00ac6> (accessed Jun. 08, 2021).
- [55] V. Agarwal, “Complete Architectural Details of all EfficientNet Models | by Vardan Agarwal | Towards Data Science,” *Towards Data Science*, May 23, 2020. <https://towardsdatascience.com/complete-architectural-details-of-all-efficientnet-models-5fd5b736142> (accessed Jun. 08, 2021).
- [56] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jan. 2018, pp. 4510–4520, doi: 10.1109/CVPR.2018.00474.
- [57] R. Agarwal, “The 5 Classification Evaluation metrics every Data Scientist must know | by Rahul Agarwal | Towards Data Science,” *Towards Data Science*, Sep. 17, 2019.

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 92 de 104

<https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226> (accessed Jun. 08, 2021).

- [58] H. M and S. M.N, “A Review on Evaluation Metrics for Data Classification Evaluations,” *Int. J. Data Min. Knowl. Manag. Process*, vol. 5, no. 2, pp. 01–11, Mar. 2015, doi: 10.5121/ijdkp.2015.5201.
- [59] C. Pere, “What are Loss Functions?. After the post on activation functions... | by Christophe Pere | Towards Data Science,” *Towards Data Science*, Jun. 17, 2020. <https://towardsdatascience.com/what-is-loss-function-1e2605aeb904> (accessed Jun. 10, 2021).
- [60] S. M. Vieira, U. Kaymak, and J. M. C. Sousa, “Cohen’s kappa coefficient as a performance measure for feature selection,” 2010, doi: 10.1109/FUZZY.2010.5584447.
- [61] R. Wirth, “CRISP-DM : Towards a Standard Process Model for Data Mining,” *Proc. Fourth Int. Conf. Pract. Appl. Knowl. Discov. Data Min.*, no. 24959, pp. 29–39, 2000.
- [62] J. Villena Román, “CRISP-DM: La metodología para poner orden en los proyectos,” *Sngular*, 2016. <https://www.sngular.com/es/data-science-crisp-dm-metodologia/> (accessed Nov. 12, 2020).
- [63] PapersWithCode, “ImageNet Benchmark (Image Classification) | Papers With Code,” *Papers With Code*. <https://paperswithcode.com/sota/image-classification-on-imagenet>

	DOCUMENTO FINAL PROYECTO DE GRADO	EDO – 02
		Versión 1
		Vigencia: 30/06/2021
		Página 93 de 104

(accessed Jun. 19, 2021).

- [64] P. Sharma, “Video Classification | Video Classification Model In Python,” *Analytics Vidhya*, Sep. 03, 2019. <https://www.analyticsvidhya.com/blog/2019/09/step-by-step-deep-learning-tutorial-video-classification-python/> (accessed Jun. 23, 2021).
- [65] Keras, “SeparableConv2D layer,” *Keras API reference*, 2021. https://keras.io/api/layers/convolution_layers/separable_convolution2d/ (accessed Jun. 08, 2021).