UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

**TRABAJO FIN DE MASTER EN INGENIERÍA INDUSTRIAL**

# Diseño de un modelo compuesto de arquitecturas de red neuronal convolucional y recurrente para descripción de vídeo en entornos outdoor/indoor


AUTOR: ALEJANDRO GOLFE SAN MARTÍN

TUTOR: BEATRIZ ANA TRÉNOR GOMIS

COTUTOR: ADRIÁN COLOMER GRANERO


**Curso Académico: 2020-21**

# RESUMEN

Actualmente existen en todo el mundo un gran número de personas invidentes. Estas personas tienen grandes dificultades para desempeñar actividades cotidianas que son sencillas para cualquier individuo sano, esto es debido a que la vista es uno de los principales sentidos que posee el ser humano. El presente trabajo consiste en aplicar las técnicas más recientes de inteligencia artificial con el objetivo de crear un dispositivo de visión por computador que permita procesar lo que ocurre en cualquier entorno y generar una descripción textual del mismo. Gracias a la ayuda de un sintetizador de voz, se generará un fichero de audio a partir de las descripciones recibidas.

De manera más específica, en este trabajo se diseña, implementa y valida un modelo compuesto por arquitecturas de redes neuronales convolucionales y recurrentes para descripción de video indoor/outdoor en cualquier entorno. Después de definir las métricas de evaluación necesarias, se justificará la elección del modelo óptimo para ser embebido en un dispositivo portable permitiendo evaluar el rendimiento del modelo implementado. La adaptación del modelo a un dispositivo portable estará compuesta tanto por el montaje, como por la programación y adecuación del problema a la capacidad del dispositivo portable empleado.

Finalmente, se evalúa tanto el rendimiento y resultados que ofrece el modelo implementado como los conocimientos adquiridos por el alumno para el desarrollo del proyecto. Por último, se proponen posibles líneas de investigación para mejorar los resultados obtenidos.

Palabras clave:

- Visión artificial
- Deep learning
- Invidentes
- Inteligencia Artificial
- Python
- Redes neuronales
- Raspberry Pi
- TensorFlow

# ABSTRACT

There are currently a large number of blind people around the world. These people have great difficulties to perform daily activities that are simple for any healthy individual, since sight is one of the main senses possessed by human beings. The present work consists of applying the most recent techniques of artificial intelligence with the objective of creating a computer vision device that allows processing what happens in any environment and generating a textual description of it. Thanks to the help of a voice synthesizer, an audio file will be generated from the descriptions received.

More specifically, this work designs, implements and validates a model composed of convolutional and recurrent neural network architectures for indoor/outdoor video description in any environment. After defining the necessary evaluation metrics, the choice of the optimal model to be embedded in a portable device will be justified to evaluate the performance of the implemented model. The adaptation of the model to a portable device will consist of the assembly, programming, and adaptation of the problem to the capacity of the portable device used.

Finally, the performance and results offered by the implemented model are evaluated, as well as the knowledge acquired by the student for the development of the project. Finally, possible lines of research are proposed to improve the results obtained.

Keywords:

- Artificial vision
- Deep learning
- Blind people
- Artificial Intelligence
- Python
- Neural networks
- Raspberry Pi
- Keras

# RESUM

Actualment existeixen a tot el món un gran nombre de persones invidents. Aquestes persones tenen grans dificultats per a exercir activitats quotidianes que són senzilles per a qualsevol individu sa, això és pel fet que la vista és un dels principals sentits que posseeix l'ésser humà. El present treball consisteix a aplicar les tècniques més recents d'intel·ligència artificial amb l'objectiu de crear un dispositiu de visió per computador que permeta processar el que ocorre en qualsevol entorn i generar una descripció textual d'aquest. Gràcies a l'ajuda d'un sintetitzador de veu, es generarà un fitxer d'àudio a partir de les descripcions rebudes.

De manera més específica, en aquest treball es dissenya, implementa i valida un model compost per arquitectures de xarxes neuronals convolucionals i recurrents per a descripció de vídeo indoor/outdoor en qualsevol entorn. Després de definir les mètriques d'avaluació necessàries, es justificarà l'elecció del model òptim per a ser embegut en un dispositiu portable permetent avaluar el rendiment del model implementat. L'adaptació del model a un dispositiu portable estarà composta tant pel muntatge, com per la programació i adequació del problema a la capacitat del dispositiu portable empleat.

Finalment, es evalua tant el rendiment i resultats que ofereix el model implementat com els coneixements adquirits per l'alumne per al desenvolupament del projecte. Finalment, es proposen possibles línies d'investigació per a millorar els resultats obtinguts.

Paraules clau:

- Visió artifificial
- Deep learning
- Invidents
- intel·ligència Artificial
- Python
- Xarxa neuronal
- Raspberry Pi
- Keras

# Índice Global

# ÍNDICE DE ILUSTRACIONES

# ÍNDICE DE TABLAS

**TABLAS MEMORIA**



**TABLAS PRESUPUESTO**

# I. MEMORIA

# 1 Introducción

## 1.1 Justificación y contexto del trabajo

Desde los orígenes del ser humano, el sentido de la vista es uno de los más importantes a la hora de garantizar la supervivencia del individuo. Actualmente, se estima que 188,5 millones de personas tienen una deficiencia visual moderada, 217 millones presentan una deficiencia grave y 36 millones son completamente invidentes [1].Las personas que presentan una carencia total de visión pueden encontrar grandes problemas a la hora de realizar tareas cotidianas muy sencillas para cualquier individuo sin discapacidad. Este problema se distribuye de manera desigual según la edad, siendo mayor la incidencia en personas mayores de 50 años. Los factores de riesgo asociados a esta discapacidad que son considerados comunes son la edad, el género y la condición socioeconómica.

Se estima que el coste que conlleva la ceguera en España es cercano a los 360 millones de euros, cifra que no considera los costes asociados a la Miopía Patológica por su complejidad de cálculo. Según el informe de la ceguera en España, el coste derivado de la ceguera para cada discapacitado es de 5.100€ anuales [2]. La discapacidad visual plantea a los que la padecen una serie de barreras sociales que trascienden la propia discapacidad. Todavía existe un largo recorrido hasta que las personas con problemas de ceguera puedan llevar a cabo una vida normal.

Las consecuencias que se pueden derivar de esta discapacidad son innumerables, como depresión, aislamiento social y segregación por el elevado coste económico anual. Cualquier ayuda que permita a la persona con ceguera llevar una vida más parecida a lo que se considera normal, no solo supone un ahorro económico, sino que permite a estos individuos llevar una mejor calidad de vida. Con el fin de promover la integración de las personas con discapacidad visual se estudia desde el punto de vista de la ingeniería de qué manera se pueden aplicar los conocimientos técnicos y tecnología actual para mejorar la vida de estas personas.

De este problema surge la necesidad de crear un dispositivo que sea capaz de procesar el vídeo captado de un entorno en cierto intervalo temporal para generar una descripción de texto que permita al usuario entender que está sucediendo a su alrededor. El avance de la tecnología ha abierto la posibilidad de emplear sistemas de inteligencia artificial para el procesamiento y comprensión de escenas, permitiendo realizar tareas de clasificación y detección de objetos de esta con el fin de contextualizarla.

El presente trabajo tiene como finalidad implementar distintas arquitecturas de redes neuronales artificiales que permitan el procesamiento en tiempo real del entorno que rodea a un individuo y su descripción mediante una sentencia de texto, generando adicionalmente un fichero de audio con la lectura de la sentencia generada.

## 1.2 Planificación del trabajo

La primera fase del proyecto consiste en identificar el problema que se quiere resolver y crear un título, resumen, palabras claves y objetivos del proyecto que se va a desarrollar.

Una vez que se ha definido claramente la línea de trabajo se destinará una gran parte de tiempo a investigar y aprender sobre las últimas tendencias en descripción de video mediante inteligencia artificial.

Después de estudiar la documentación necesaria, se seleccionará la arquitectura de red que se considere más adecuada y se llevará a cabo un análisis de distintas bases de datos públicas, para finalmente escoger la que mejor se ajuste a esta aplicación.

Terminado lo anterior, se procederá a la fase de diseño, desarrollo e implementación del algoritmo basado en redes neuronales y que será alimentado por la base de datos seleccionada. El resultado de esto será un modelo de conocimiento capaz de procesar los fotogramas extraídos del video captado, generando a la salida una descripción textual del mismo gracias al conocimiento almacenado en la arquitectura compuesta por redes neuronales convolucionales y recurrentes. Finalmente, dicha descripción textual se introducirá en un generador de discurso para generar el archivo de audio correspondiente con la lectura en voz alta de la descripción textual del entorno generada.

Una vez terminado y entrenado el modelo, este se embeberá en un dispositivo Raspberry Pi 4 para llevar a cabo el reconocimiento de vídeo en tiempo real. Cuando la implementación sea satisfactoria, se procederá a evaluar los resultados que ofrece el modelo.

Después de todo el proceso mencionado anteriormente, se procederá a redactar la memoria del proyecto y posteriormente a su defensa ante el tribunal.

El proceso descrito se resume en la Ilustración 1.

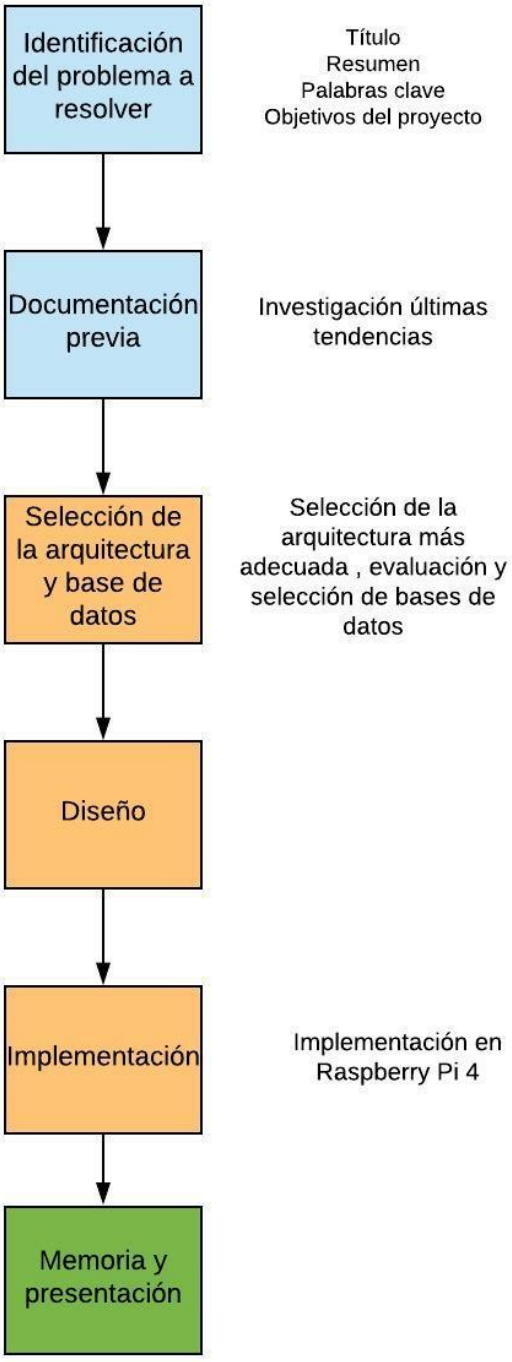



*Ilustración 1. Esquema de la planificación del proyecto.*

## 1.3 Estado del arte

Actualmente, gracias a las plataformas de vídeo e imágenes, se cuenta con una gran cantidad de información digitalizada. Muchas actividades que resultan sencillas para los seres humanos suponen un reto a la hora de implementarlas en cualquier dispositivo con capacidad de cómputo. La descripción de vídeo es una de estas actividades, ya que el ser humano no tiene problemas para describir lo que ve en su entorno, pero dotar a una máquina de la capacidad para realizar esta tarea es un problema de gran complejidad.

La tecnología de visión por computador está adquiriendo gran relevancia actualmente. La capacidad de proporcionar a una máquina la habilidad de procesar información visual mediante procesos de aprendizaje, permite llevar a cabo tareas de resolución de problemas en múltiples disciplinas. Uno de los campos es el conocido como *Image and video captioning* en inglés, el cual engloba los procesos para convertir la información almacenada en una imagen en una descripción en lenguaje comprensible para los seres humanos. Esta disciplina tiene múltiples aplicaciones de clasificación, resumen de secuencias o la vigilancia por vídeo, entre muchas otras.

Para generar automáticamente descripciones de video, se deben seguir las siguientes fases:

- Identificación de objetos
- Reconocimiento de la actividad
- Generación de la descripción textual

Los primeros trabajos en este campo se centraban en la descripción de imagen [3]–[6], luego estos métodos se extendieron para descripción de video [7]–[11] ,aunque su aplicación se limitó a entornos muy concretos (por ejemplo, cocina), limitándose a un vocabulario y entornos muy específicos. Este tipo de entornos impide que el modelo diseñado sea incapaz de generalizar fuera del ámbito para el que ha sido pensado. La capacidad de que un modelo permita generalizar para cualquier entorno viene limitada por el vocabulario necesario y por la falta de bases de datos con información anotada.

Los primeros trabajos que se encuentran en la literatura son los siguientes [12]–[14], en los cuales se combinan procedimientos de reconocimiento visual a partir de fotogramas de vídeo para descripción de video. Concretamente, se entrenan distintos clasificadores para llevar a cabo identificación de objetos, actividades y escenas individualmente. A continuación, empleando la probabilidad de clasificación junto con el conocimiento del lenguaje empleado, se realizan las mejores predicciones de sujeto, verbo, objeto y escena.

La aparición del Deep Learning (aprendizaje profundo) ha revolucionado los ámbitos de visión por computador. El empleo de redes neuronales convolucionales ha demostrado ser un método eficaz para identificar objetos [15]. Las redes neuronales recurrentes han probado su eficacia en procesos de modelo de secuencias procesando lenguaje natural y en traducción automática [16].

A pesar de todo, la descripción de vídeo no ha sido objeto de tanta investigación. Los autores en [17] proponen el uso de una red neuronal convolucional junto con una recurrente tipo Long Short Term Memory para extraer información visual de las imágenes y describir las secuencias respectivamente. En [17] se propone un entrenamiento que consiste en pares imagen-descripción para llevar a cabo el aprendizaje. Posteriormente, empleando técnicas de transferencias de conocimiento y el uso de bases de datos con anotaciones de pares imagen-descripción se dota al modelo diseñado de la capacidad para describir secuencias de imágenes. Tomando como punto de partida el anterior trabajo, en [18] Venugopalan et al. cambiaron el diseño de la arquitectura de red para capturar la dependencia de una secuencia de imágenes y la secuencia de palabras que conforma la descripción. A este sistema se le incluyó conocimiento lingüístico con el fin de mejorar la calidad gramatical y descriptiva de los vídeos [19].

En [20] se muestran las distintas arquitecturas empleadas en la literatura. Existen tres tipos de arquitecturas que destacan en la actualidad:

- CNN + RNN (Convolucional y recurrente)
- RNN-RNN (Recurrente en ambas fases de la descripción de vídeo)
- Redes de refuerzo profundas (Área todavía en investigación.

Respecto a la primera opción, debido al éxito de las redes neuronales convolucionales en el ámbito de la visión artificial, estas son las más empleadas ya que además son más sencillas que otro tipo de arquitecturas. Los primeros trabajos en los que se emplearon las redes neuronales profundas para la descripción de vídeo fueron propuestos por Donahue et al. [21]. El uso de redes neuronales recurrentes, en particular LSTM, en la decodificación de las imágenes previamente codificadas por la red neuronal convolucional, fue introducido por Rohrbach et al. [22]. Los últimos avances en esta tecnología se muestran en [23] y consisten en emplear varias capas de redes neuronales recurrentes tipo LSTM, en concreto una para la codificación *one-hot* de la frase de entrada, lo que permite generar frases de longitud variable. La última representación de la capa oculta de la codificación se hace pasar por una capa de decodificación que genera una frase palabra por palabra en cada salto temporal.

En cuanto a la propuesta de emplear redes neuronales recurrentes tanto en la codificación de las imágenes como en la codificación de frases textuales, estas propuestas no son tan populares. Destaca el trabajo de Srivastava et al. [24], donde empleaba una red LSTM para codificar el contenido de las imágenes y extraer las características para a continuación enviar la información resultante a una capa LSTM de decodificación. Yu et al. [25] propuso un modelo similar para esta tarea de descripción de vídeo, con la diferencia de que empleó una configuración jerárquica con unidades de múltiples puertas para la generación de descripciones. La salida de este decodificador se enviaba a un generador de párrafos que modelaba la dependencia temporal de las descripciones generadas. Estos modelos demostraron ser poco eficientes para actividades con objetos de pequeño tamaño.

Las redes de refuerzo profundas han demostrado superar en muchos ámbitos la capacidad del ser humano simplemente aplicando técnicas de aprendizaje por refuerzo y penalización. Estos avances son populares desde 2013 gracias a Google Deep Mind [26] [27]. Debido a que estos mecanismos de aprendizaje no tienen una función de coste definida, su desempeño es difícil de evaluar.

Xwang et al. [28] propuso un modelo de aprendizaje profundo por refuerzo para descripción de vídeo, que consistía en un modelo codificador decodificador. La parte que codifica la información contenida

en las imágenes utilizaba el conocimiento presente en la base de datos ResNet-152 [29]. La decodificación la realiza una red LSTM compuesta por un tramo de bajo nivel y uno de alto nivel. En 2018 Chen et al. [30] en el cual seleccionaba los fotogramas clave del vídeo para describir un vídeo completo con el objetivo de minimizar el ruido y el cómputo innecesario. El criterio para escoger estos fotogramas claves era que el contenido visual fuera máximo y la discrepancia textual de la descripción fuera mínima. En este trabajo, se estima que una selección de seis u ocho fotogramas eran suficientes para describir un vídeo completo. Estos métodos siguen en continua expansión y se espera que pronto supongan una gran revolución en el campo de la descripción de vídeo.

A continuación, se muestran algunos ejemplos de dispositivos similares que se encuentran en el mercado. El dispositivo Orcam MyEye Pro (Ver Ilustración 2) es un conocido dispositivo de asistencia para personas con discapacidad visual, permite a sus usuarios leer el periódico, leer libros, reconocer los rostros de los seres queridos e incluso asistir para realizar la compra. Esta tecnología tiene el tamaño necesario para adaptarse a cualquier marco de gafas y cuenta con decenas de miles de usuarios en todo el mundo, siendo empleado en más de 40 países y contando con lenguaje en 20 idiomas. Su precio actual es de 4750,00€ por lo que no todo el que lo necesita es capaz de permitírselo. Se recuerda que las personas con discapacidad visual normalmente tienen más problemas a la hora de acceder al mundo laboral.


*Ilustración 2. Dispositivo Orcam MyEye Pro* [31]

Otro dispositivo presente en el mercado que puede soportar la tecnología de descripción de vídeo son las Google Glass. Este proyecto fue lanzado en 2012 y resultó un éxito mediático debido a que se trataba de un producto futurista que resolvía problemas que en aquel momento resultaban de ciencia ficción. Aun así, resultó un fracaso a nivel de producto de consumo y a pesar de su carácter innovador, quedaron en el olvido. Después de unos años, Google ha decidido seguir apostando por su proyecto introduciendo un nuevo software de inteligencia artificial desarrollado por Envision. Esta aplicación fue premiada en 2019 en los Google Play Awards y permite explorar el entorno proporcionando la información en forma de salida de audio. Este software permite leer carteles y textos, describir lo que hay a nuestro alrededor y reconocimiento facial entre otras múltiples funciones. El coste de estas gafas es inferior al dispositivo Orcam MyEye Pro, siendo este de 1200$, aunque sigue creando una brecha económica, por lo que muchas personas con discapacidad visual no podrán tener acceso a ellas.

*Ilustración 3. Google Glass* [32].

## 1.4 Objetivos del trabajo

El objetivo del presente trabajo es integrar los conocimientos adquiridos durante la formación del alumno en inteligencia artificial y programación para diseñar un modelo basado en redes neuronales artificiales que permita procesar el entorno y describir textualmente a las personas invidentes lo que ocurre a su alrededor. Este trabajo es un apéndice de un proyecto más grande que pretende llevar a un dispositivo hardware el modelo diseñado.

Se implementarán distintas arquitecturas de redes neuronales artificiales que conformarán un modelo capaz de describir textualmente lo que ocurre en un conjunto de imágenes extraídas de un video.

Las actividades que se desarrollarán en este proyecto para validar el resultado del modelo desarrollado son las siguientes:

- Recopilación de información sobre el estado del arte de la descripción de video empleando modelos de redes neuronales artificiales.
- Recopilar información sobre bases de datos existentes para entrenar el modelo diseñado, ya que debe detectar cualquier tipo de entorno.
- Análisis de los requerimientos computacionales de la aplicación y selección del entorno de programación adecuado.
- Diseñar y desarrollar la arquitectura de red neuronal más idónea para el problema que se plantea en este proyecto.
- Implementar el modelo predictivo resultante en un dispositivo que permita la captación, procesamiento de imagen y generación del fichero de voz.
- Evaluar los resultados obtenidos y comparar su eficacia con otros modelos.

La elaboración de este proyecto requiere tanto de la formación obtenida por el alumno como de un gran componente de formación autodidacta, ya que la profundidad del conocimiento necesario para realizar este tipo de modelos requiere de un mayor grado de especialización.

# 2 Marco teórico

En esta sección se agrupa la información necesaria para comprender los fundamentos teóricos que se encuentran detrás del algoritmo implementado en este trabajo. El objetivo es ofrecer al lector una vista general de cómo funcionan las distintas arquitecturas implementadas y su fundamento teórico.

## 2.1 Introducción

El concepto de *Machine Learning* está íntimamente relacionado con el concepto de inteligencia artificial. *Machine Learning*, o aprendizaje automático, se refiere a la capacidad de un software o dispositivo de aprender a realizar cierta tarea a través del ajuste de algoritmos en base a unos datos de entrada.

Esta disciplina se engloba en las conocidas como ciencias de computación y sirve para llevar a cabo la creación de sistemas con capacidad de aprender por sí mismos. Gracias a esta tecnología, se pueden automatizar tareas para reducir la intervención del ser humano. Esto permite gestionar volúmenes de información elevados que serían imposibles de procesar por un cerebro humano con la misma eficacia.

La razón de que esta disciplina se conozca como "aprendizaje", es porque el sistema es capaz de identificar patrones complejos en los datos de entrada para un gran número de parámetros. El hecho de que se conozcan a estos algoritmos como de aprendizaje autónomo, es debido a que ellos mismos identifican y extraen los patrones, pero debe ser un ser humano el que inicie dicho proceso de aprendizaje.

Existen cuatro tipos diferentes de *Machine Learning*:

- **Aprendizaje supervisado**: este tipo de aprendizaje funciona con datos de entrenamiento del sistema diseñado. Estos datos de entrenamiento se proporcionan al sistema y vienen anotados con etiquetas. La finalidad de esto es que el algoritmo sepa distinguir las características propias de cada etiqueta, para que posteriormente se le proporcione al sistema datos de entrada sin etiquetar y sea capaz de clasificarlos. El modelo de redes neuronales diseñado en este proyecto se engloba dentro de este tipo de aprendizaje.

- **Aprendizaje no supervisado**: a diferencia del tipo anterior, los datos no vienen etiquetados y, por tanto, queda en manos del modelo discernir los patrones que permiten diferenciar los datos de entrada. Este tipo de aprendizaje es el que realiza en muchas ocasiones el ser humano.

- **Aprendizaje semi-supervisado**: es un tipo de aprendizaje que emplea datos etiquetados y no etiquetados. Normalmente se cuenta con un reducido número de datos etiquetados junto a un elevado número de datos sin etiquetar. Este aprendizaje es un paso intermedio entre el supervisado y no supervisado. Algunos investigadores han comprobado que el uso conjunto de estos datos ofrece un incremento de rendimiento significativo [33].

- **Aprendizaje por refuerzo**: estos sistemas aprenden a partir de la experiencia a través de un proceso de recompensa y penalización. Se registran las acciones llevadas a cabo y se evalúa su calidad. A medida que el sistema vaya aprendiendo, identificará los valores correctos gracias a su experiencia. Es imprescindible para este tipo de aprendizaje definir correctamente la función de evaluación de la solución adoptada. Una de sus ventajas es que permite comenzar a operar sin introducir una gran cantidad de datos.

El aprendizaje profundo, o *Deep Learning*, es un conjunto de algoritmos que se engloban dentro del aprendizaje automático, cuyo objetivo es modelar abstracciones de gran volumen de datos empleando arquitecturas computacionales que permiten realizar transformaciones no lineales [34].

Dentro del aprendizaje profundo se encuentran las arquitecturas de redes neuronales profundas, las cuales se han utilizado para aplicaciones de visión por computador. En tareas como reconocimiento automático del habla e identificación de señales de audio y música también han demostrado gran eficacia.

No existe una definición oficial de aprendizaje profundo, aunque todas las existentes tienen el eje común de que este tipo de aprendizaje son una serie de algoritmos que se encuentran dentro del aprendizaje automático. De este eje común parten distintas definiciones, en función de las características en las que se centran, se muestran a continuación algunos ejemplos:

- Aprendizaje de numerosas características y representaciones de datos. Las características se jerarquizan en función de su nivel.
- Emplear una batería de capas de procesamiento no lineal para extraer y transformar variables. Las capas van conectadas sucesivamente, la salida de una es la entrada de la siguiente. Se puede emplear tanto aprendizaje supervisado como no supervisado, cuyo objetivo es modelizar datos e identificar patrones.
- Llevar a cabo el aprendizaje de una gran cantidad de niveles de representación en función de los distintos niveles de abstracción.

Este tipo de aprendizaje se puede encontrar en sistemas de computación en la nube, empresas como Amazon, Azure, IBM o Google ofrecen este tipo de servicios.

## 2.2 Red neuronal biológica

La neurona como unidad discreta que conforma el sistema nervioso fue una idea introducida por Santiago Ramón y Cajal [35]. En su trabajo de investigación las definió como unidades que se conectan formando conexiones especializadas, lo que se conoce como red neuronal.

La función de las neuronas es transmitir información de manera rápida y precisa a otras células. Esta información tiene la forma de señales bioeléctricas conocidas como impulsos nerviosos. La estructura de la neurona es la que se muestra en la Ilustración 4.

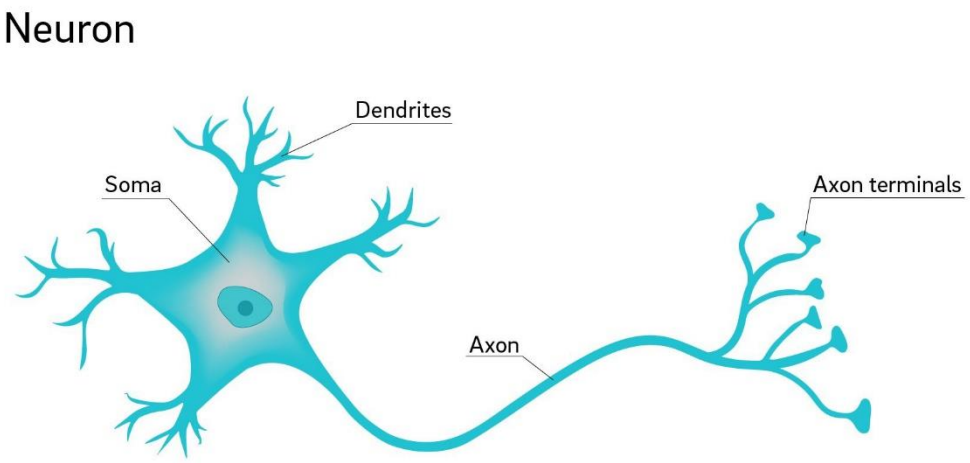



*Ilustración 4. Esquema simplificado de una red neuronal biológica* [36].

El impulso nervioso es recibido por las dendritas y transmitido por toda la neurona a través del axón, llegando a los botones terminales que transmiten el impulso a la neurona siguiente. Estas conexiones entre neuronas reciben el nombre de sinapsis. En la Ilustración 5 se muestra un ejemplo de la conexión entre dos neuronas a través del botón terminal.

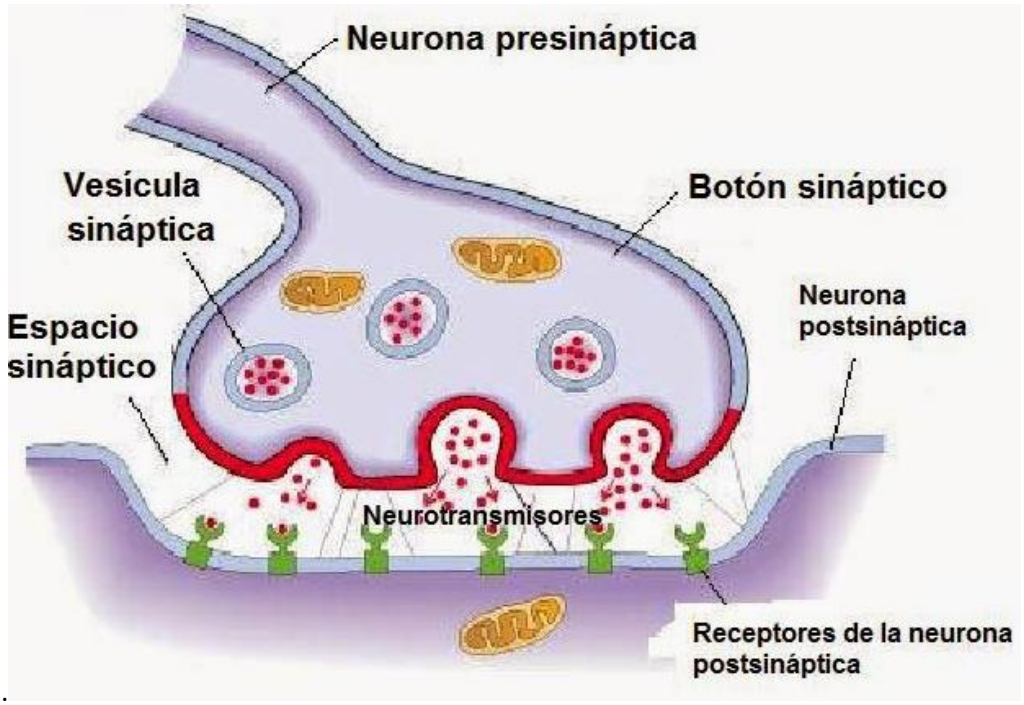



*Ilustración 5. Esquema de los elementos que componen la sinapsis neuronal* [37].

En esta conexión se produce lo que se conoce como salto sináptico. Se liberan neurotransmisores por parte de la neurona presináptica y son recibidos por la postsináptica. Fruto de esta propagación se crea el estímulo en la siguiente neurona, este proceso se repite sucesivamente hasta que el impulso nervioso alcanza el lugar deseado por el sistema nervioso.

Este proceso biológico ha inspirado el modelo de programación que se explica en este proyecto. La cantidad de neurotransmisores que se asignan a una conexión sináptica es fruto del aprendizaje del sistema nervioso. La programación de redes neuronales pretende imitar este tipo de aprendizaje para crear modelos generales que sean capaces de extraer características y patrones de datos de entrada como realizaría un cerebro humano.

Al igual que el número de conexiones que se encuentran en el sistema nervioso es elevado, un modelo de programación eficaz para resolver tareas complejas puede llevar asociado grandes requerimientos computacionales. Actualmente la tecnología ha avanzado mucho en este campo y algunas compañías como Google han desarrollado hardware específico para este tipo de aplicaciones, lo que ha provocado un gran salto en potencia de cálculo y eficiencia. Esta revolución ha permitido realizar algunas tareas que antes se descartaban por la poca viabilidad que ofrecía su coste computacional.

## 2.3 Red neuronal artificial

Las redes neuronales artificiales son un modelo computacional inspirado en el comportamiento de las redes neuronales biológicas [38]. Están compuestas por un conjunto de neuronas artificiales conectadas entre sí para llevar a cabo la transmisión de señales. La señal atraviesa la red neuronal artificial produciendo un valor a la salida de ésta.

Estos sistemas tienen como objetivo reproducir la capacidad que tienen los seres humanos para interactuar con su entorno y aprender de él para resolver los problemas de la misma forma que un cerebro humano. La gama de tareas que se han resuelto con estos sistemas es muy amplia, tienen gran importancia en sistemas de reconocimiento de voz y en visión artificial.

En este trabajo se han aplicado dichos sistemas de redes neuronales para realizar la extracción de características de imágenes contenidas en un vídeo, y asociar dichas características a una descripción de texto. La combinación de extracción de características y generación de una descripción textual ordenada secuencialmente ha requerido combinar distintas arquitecturas de redes neuronales, como son las redes neuronales convolucionales y las redes neuronales recurrentes.

Las redes neuronales artificiales se componen de cuatro elementos básicos:

- **Neuronas**: unidad básica de la red neuronal artificial que recibe información externa y proporciona un valor de salida.
- **Conexiones**: unión entre neuronas que permite el envío de señales que contienen la información.
- **Pesos sinápticos**: valores asociados a las conexiones que representan la intensidad de interacción entre la neurona origen y la neurona destino.
- **Funciones de activación**: función encargada de proporcionar un valor a la salida a partir de un valor de entrada. Normalmente se emplean funciones cuya derivada es simple para disminuir el coste computacional.

En la Ilustración 6 se distinguen tres tipos de capas que componen la red neuronal artificial, las cuales se describen a continuación:

- **Capa de entradas**: capa que recibe la información del exterior. En el presente trabajo será la capa que contendrá el video descompuesto en imágenes del entorno.
- **Capa de ocultas**: son las capas intermedias entre la entrada y la salida que tienen como función dotar de no linealidades al sistema. El número de niveles que pueden componer dicha capa es muy variable y sus interconexiones determinan la topología de red empleada.
- **Capa de salida**: son las capas encargadas de transmitir la información obtenida del conjunto de la red neuronal artificial al exterior.

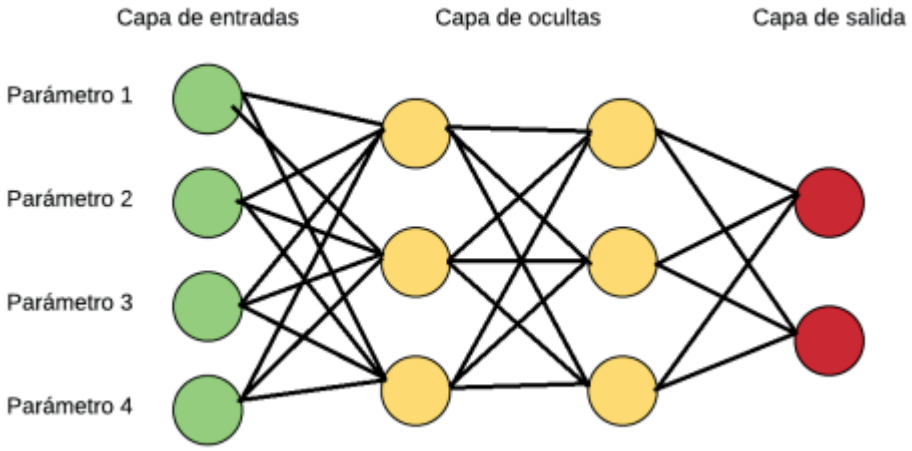



*Ilustración 6. Esquema básico de una red neuronal artificial* [39].

Tanto el número de neuronas como sus conexiones entre ellas afectan directamente al coste computacional durante la etapa de entrenamiento (o creación) del modelo. Existen técnicas para reducir el número de conexiones, lo que se conoce como *dropout* [40]. Este tipo de técnicas previenen problemas como el *overfitting* de la red, las consecuencias de este problema son que el modelo no es capaz de generalizar y solamente reconoce los elementos con los que ha sido entrenado.

En cuanto a la implementación del modelo, la fase inicial corresponde a la fase de entrenamiento. En esta etapa se realiza un ajuste de los pesos sinápticos de las conexiones mediante un algoritmo que minimice el error entre la salida de la red y el dato real. Los procedimientos en los que se tiene el dato verdadero asociado a las entradas de la red se conocen como procedimientos de aprendizaje supervisado, en este trabajo se va a emplear dicho tipo de aprendizaje ya que cada fotograma del vídeo se asocia a una descripción de texto previamente etiquetada. De los algoritmos empleados para minimizar el error en la capa de salida destaca el método *stochastic gradient descent* junto a la técnica de *backpropagation* para retropropagar dicho error hacia capas predecesoras de la red.

Una vez entrenado el modelo, se somete a la etapa de validación o test. En esta fase se comprueba el comportamiento del modelo diseñado frente a distintos valores de entrada. Si se ha realizado correctamente la etapa de entrenamiento, el modelo será capaz de generalizar y proporcionar respuestas correctas ante *inputs* previamente desconocidos.

La diferencia de la etapa de validación o test con la de entrenamiento es que no se modifican los pesos sinápticos de las conexiones que contiene la red neuronal. Es imprescindible que la fase de entrenamiento se realice con una muestra representativa de la población de datos que se está estudiando, ya que si no el modelo resultará ineficaz a la hora de encontrar la respuesta correcta.

## 2.4 Ventajas e inconvenientes de las redes neuronales artificiales

De entre todas las ventajas que tienen este tipo de algoritmos de inteligencia artificial, destacan las siguientes:

- **Aprendizaje:** los modelos de redes neuronales artificiales tienen la capacidad de aprender de los datos que se le proporcionan en la etapa de entrenamiento. No es necesario definir ni conocer los patrones que diferencian el conjunto de datos que se les proporciona. Esta característica es especialmente útil ya que permite al algoritmo encontrar por él mismo patrones que probablemente el ser humano sería capaz de encontrar debido al volumen de información. Esta propiedad de aprendizaje permite a estos modelos adaptarse a nuevas condiciones que se le introduzcan.

- **Simplicidad de implementación:** la existencia de numerosas librerías y trabajos de investigación en este campo permiten al desarrollador/investigador implementar modelos de redes neuronales prescindiendo del desarrollo a nivel bajo.

- **Tolerancia a fallos**: la redundancia de la información contenida en las redes neuronales artificiales permite que, en caso de error o daño en la información, el modelo pueda seguir funcionando correctamente.

- **Flexibilidad**: este tipo de estructuras pueden aceptar ligeras variaciones en la información de entrada sin modificar significativamente la respuesta.

Respecto a las desventajas, las más importantes son:

- **Coste computacional:** para grandes tareas con un gran volumen de información y parámetros que entrenar, el coste computacional puede ser elevado. Este problema puede conllevar que sea necesario realizar una inversión en equipos con mayor potencia de cálculo, aunque existen recursos en la red que permiten utilizar entornos de ejecución remotos con la potencia deseada.

- **Ausencia de reglas a priori:** no existe unas reglas establecidas que definan estrictamente el número de parámetros que se debe emplear o la arquitectura de red necesaria. En muchos casos depende de la experiencia del desarrollador, de la consulta de trabajos previos y de la experimentación, el determinar ciertos aspectos de la red.

- **Modelo estadístico alternativo:** en muchos problemas se plantea la duda de si un modelo estadístico sería más eficaz que un modelo de redes neuronales artificiales. Dependiendo del problema a resolver puede resultar más conveniente una alternativa u otra.

## 2.5 Redes neuronales convolucionales

Tal y como se ha mencionado en la introducción del trabajo, el modelo que se ha implementado está compuesto por una arquitectura de red neuronal convolucional y recurrente. En este apartado se explican los fundamentos teóricos de la arquitectura convolucional y su utilidad en esta aplicación.

Las redes neuronales convolucionales constan de neuronas que corresponden a campos receptivos de manera semejante a las presentes en la corteza visual de los seres vivos. Esta arquitectura de red es una derivación del perceptrón multicapa, pero su aplicación en matrices bidimensionales las hace realmente eficaces para tareas clasificación, segmentación de imágenes y cualquier otra tarea de visión artificial [41].

La convolución de $f$ y $h$ se denota como $f * h$ y se define como la integral del producto de las dos funciones después de haber desplazado una de ellas una distancia $t$ [42]. La expresión queda de la siguiente manera:

$$(f * h)(t) = \int_{-\infty}^{\infty} f(\eta)\mathrm{h}(\mathrm{t}-\eta)\mathrm{d}\eta$$

*Ecuación 1. Ecuación principal de una convolución*

En el caso de las funciones discretas se emplea una forma simplificada de la Ecuación 1, como se muestra a continuación en la Ecuación 2:

$$f[m] * h[m] = \sum_n f[n]h[m-n]$$

*Ecuación 2. Ecuación de convolución aplicada a funciones discretas.*

El término de la Ecuación 2 que corresponde a la función $h[m]$ corresponde con lo que se conoce como *kernel*. En la práctica no se lleva a cabo la operación convolución de manera estricta, sino que se aplica la correlación cruzada, que consiste en realizar la operación convolución con el kernel fijo sin voltear. Es necesario definir correctamente los términos "*kernel*" y "filtro", ya que incluso en la literatura se emplean de manera errónea. Estos términos se definen como:

- ***Kernel***: matriz de pesos que se multiplica por la matriz de entrada con el objetivo de extraer características relevantes de la imagen. Las dimensiones de la matriz *kernel* definen el nombre del tipo de convolución empleada, por ejemplo, en convoluciones 2D la matriz *kernel* es una matriz de dos dimensiones [43].

- **Filtro**: es el resultado de la concatenación de múltiples matrices *kernel*, asignando cada *kernel* a un canal particular de la entrada. Los filtros siempre tienen una dimensión mayor que el *kernel* [43].

La aplicación del *kernel* (ver Ilustración 7) a una matriz se muestra en el ejemplo de la Ilustración 8.


*Ilustración 7. Matriz kernel para extracción de características en una convolución 2D* [44].

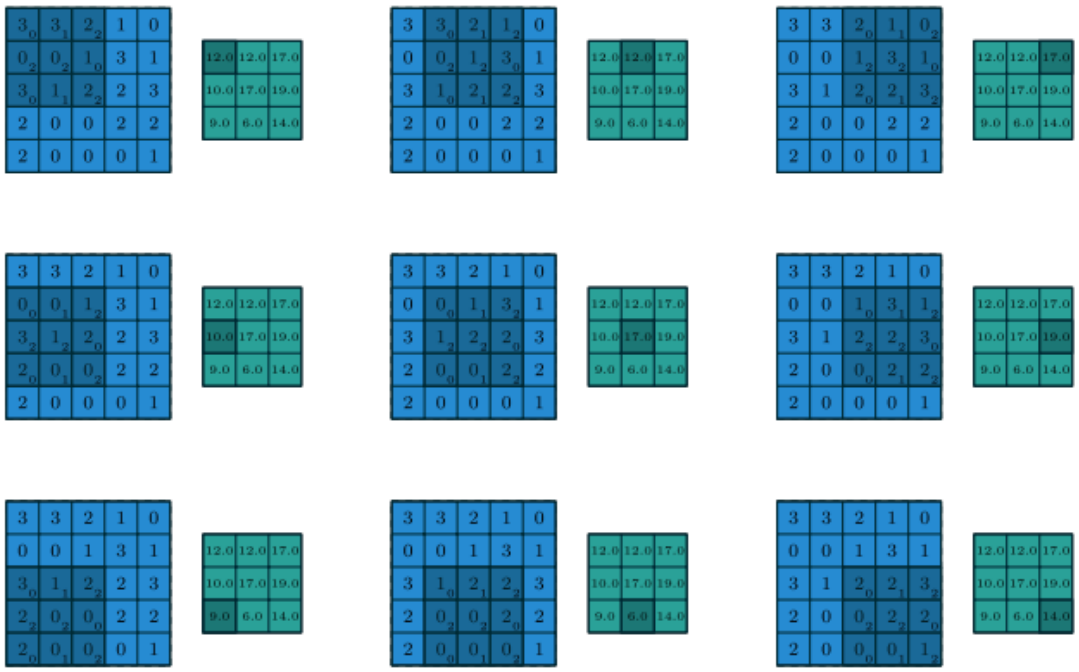



*Ilustración 8. Cálculo de los valores de salida de una convolución discreta 2D* [44].

Mediante este procedimiento se procesan las imágenes que recibe la red neuronal convolucional y se extraen las características más representativas de las mismas. Los píxeles que salta el centro de la matriz *kernel* para recorrer la matriz correspondiente a la imagen vienen dados por el parámetro conocido como *stride*. La aplicación de capas convolucionales puede provocar que se pierdan algunos píxeles en el perímetro de la imagen, para evitar este problema se suelen añadir píxeles adicionales de relleno alrededor de la imagen de entrada, esta técnica recibe el nombre de *padding*.

A la hora de definir este tipo de arquitecturas, la elección del filtro adecuado puede ser determinante sobre la eficacia de esta.

Una vez comprendida la teoría que se encuentra detrás de estos procedimientos, es conveniente definir las distintas capas que componen una red neuronal convolucional, estas son:

- Capa convolucional
- Capa de activación
- Capa de agrupación o *pooling*
- Capa de salida

La **capa convolucional** corresponde con lo descrito anteriormente, una etapa en la que se aplica un conjunto de filtros para extraer el mapa de características de cada elemento de entrada.

La **capa de activación** es la que contiene la función de activación, que se encarga de generar un valor de salida a partir de la transformación de un valor de entrada, introduciendo no linealidades en la arquitectura. La función más empleada en la literatura es la conocida como "ReLU" (ver Ilustración 9). La razón de emplear esta función en concreto es porque en procesamiento de imágenes los valores negativos no son de utilidad y si se empleara otro tipo, se modificarían las entradas a un rango muy cerrado.

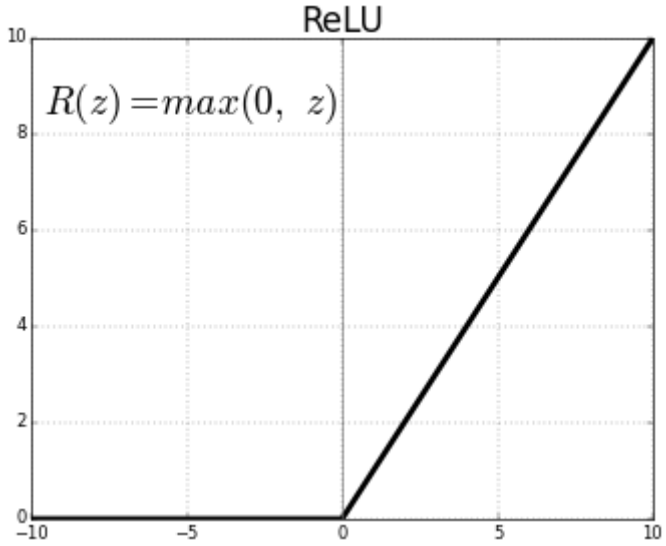



*Ilustración 9. Función de activación ReLU* [45].

Por otra parte, la **capa de agrupación** tiene como función evitar que la red alcance un tamaño que sea imposible de procesar para su entrenamiento. Esta capa lleva a cabo lo que se conoce como submuestreo escogiendo el máximo de la entrada en una ventana dada. El resultado de esta operación es un volumen de activación reducido en cuanto a sus dimensiones espaciales.

Por último, la **capa de salida** es la que recibe toda la información procesada por las anteriores etapas y proporciona la salida del modelo. En esta última etapa es donde se realiza la clasificación de los datos introducidos en la entrada. El número de neuronas a la salida es igual al número de clases en las que se pueden clasificar los datos introducidos.

## 2.6 Redes neuronales recurrentes

Las redes neuronales recurrentes (RNN) son un tipo de arquitectura que consiste en un estado oculto $h$ y una salida opcional $y$. Esta arquitectura opera con parámetros de entrada de longitud variable $x = (x_1, x_2, \dots, x_T)$. En cada instante de tiempo $t$, el valor del estado oculto $h$ se actualiza mediante la expresión dada en la Ecuación 3.

$$h_{\{t\}} = f(h_{\{t-1\}}, x_t)$$

*Ecuación 3. Actualización del valor del estado oculto de las redes neuronales recurrentes*[46].

En la Ilustración 10 se muestra una red neuronal recurrente simple en un instante de tiempo $t$. En cada paso de tiempo se transmite el valor de la función de activación.

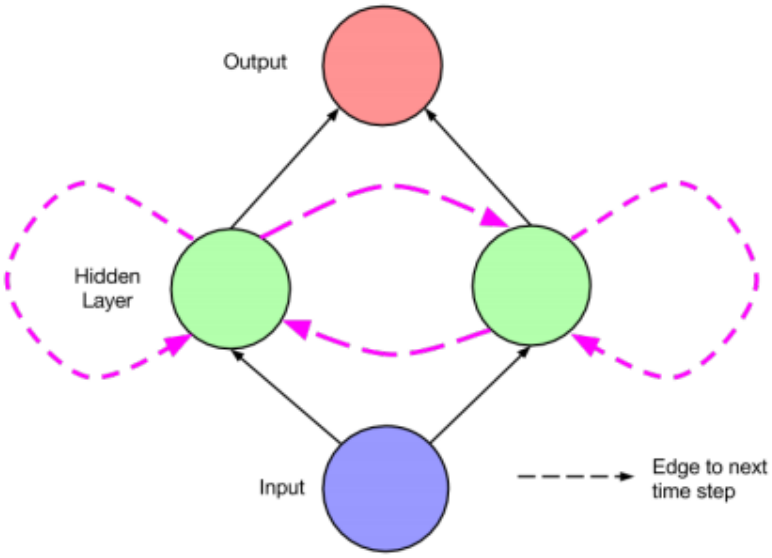



*Ilustración 10. Estructura de una red neuronal simple*[47].

Un ejemplo de la comunicación que se produce entre las estructuras de redes neuronales en instantes de tiempo diferentes sería el que se muestra en la Ilustración 11.

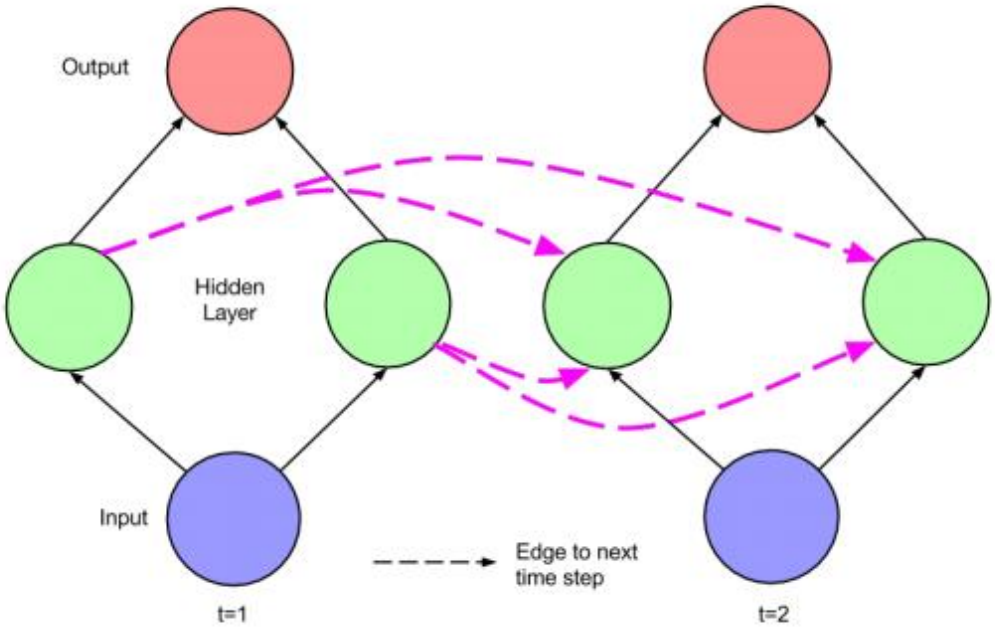



*Ilustración 11. Comunicación entre distintos pasos de tiempo en una red neuronal recurrente*[47].

El entrenamiento de las redes neuronales recurrentes puede ser bastante costoso debido a aprendizaje de dependencias de largo alcance. Uno de los principales problemas es el desvanecimiento o divergencia de los gradientes. Este problema es debido a la propagación del error a lo largo de los distintos pasos temporales, produciendo una divergencia del gradiente o una convergencia de este al valor nulo. La aparición de dicho fenómeno dependerá de si los pesos internos que conforman la red tienen valor absoluto mayor o menor que la unidad.

Una de las arquitecturas más exitosas en el campo del aprendizaje secuencial es la conocida como *Long Short-Term Memory (LSTM).* En esta estructura se introduce la célula de memoria, que sustituye a los nodos en las capas ocultas de las redes tradicionales. Cada una de estas células contiene un nodo de auto conexión con peso unitario para permitir que el gradiente atraviese distintos pasos temporales sin producirse el problema de desvanecimiento o divergencia.

La razón por la que este tipo de redes neuronales reciben este nombre es porque combinan la memoria a largo y corto plazo. Los elementos que componen las redes LSTM se describen a continuación:

- **Nodo de entrada**: este nodo $g$ recibe la información de entrada a la red y la somete a una función de activación. Normalmente esta función es una tangente hiperbólica, aunque originalmente se empleó la función sigmoide.

- **Puerta de entrada**: las puertas son un rasgo distintivo de este tipo de redes. Están compuestas por una unidad con una función sigmoide que recibe la información de entrada del estado oculto anterior. El valor de salida de esta puerta se multiplica por el valor de otro nodo, por tanto, cuando su valor es cero corta el flujo de datos de un nodo y si su valor es la unidad, deja pasar todo el flujo de información. El valor de la puerta de entrada $i$ se multiplica por el valor del nodo de entrada.

- **Estado interno**: dentro de cada célula de memoria se encuentra el nodo $s$ que presenta una activación lineal. Este estado tiene un nodo de auto conexión con un peso de valor la unidad, cuya finalidad es evitar el desvanecimiento o divergencia de gradiente. La expresión de actualización del estado interno sin puerta de olvido queda de la siguiente manera:

$$s^{(t)} = g^{(t)} * i^{(t)} + s^{(t-1)}$$

*Ecuación 4. Actualización del estado interno sin puerta de olvido.*

- **Estado de olvido**: esta puerta $f$ permite a la célula de memoria recordar u olvidar el estado interno anterior. La expresión del estado interno con esta puerta queda como se muestra en la siguiente expresión:

$$s^{(t)} = g^{(t)} * \ i^{(t)} + f^{(t)} * s^{(t-1)}$$

*Ecuación 5. Actualización del estado interno sin puerta de olvido.*

- **Puerta de salida**: el valor $v$ es el último valor que produce la célula de memoria. Es el resultado de multiplicar el estado interno $s$ por el valor de la puerta de salida $o$.

El esquema que recoge los distintos componentes de la célula de memoria se muestra a continuación en la Ilustración 12.

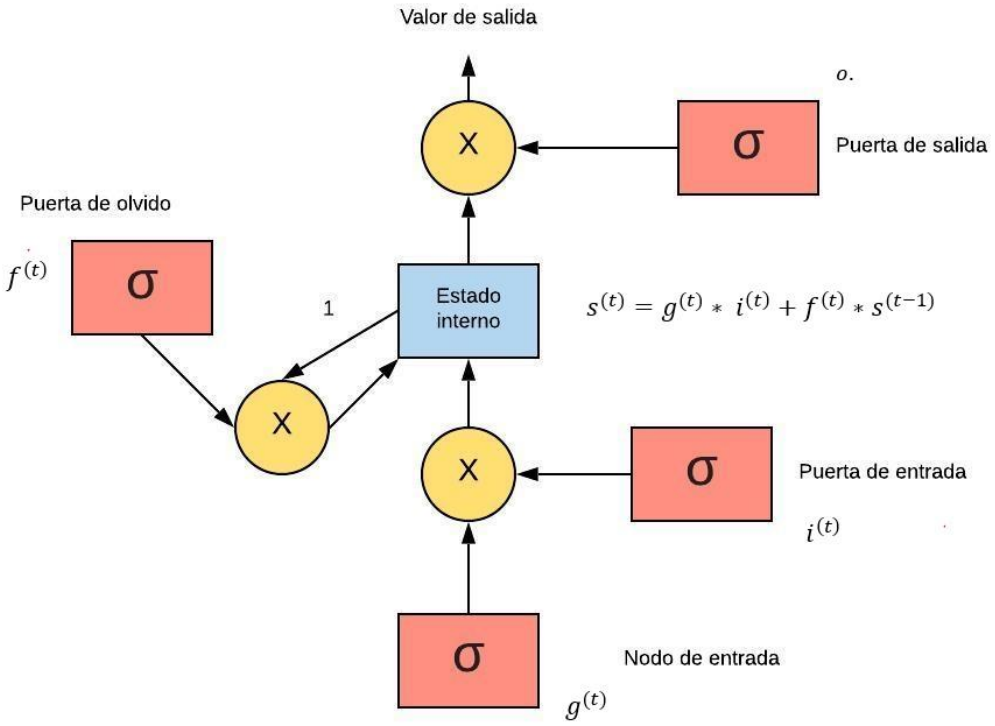



*Ilustración 12. Esquema de una célula de memoria en arquitectura LSTM.*

Las puertas de olvido no se encontraban originalmente en la arquitectura LSTM, pero su eficacia ha sido más que probada y ya se incluyen prácticamente en todas las arquitecturas modernas.

# 3 Material empleado para el desarrollo del proyecto

En este capítulo se muestra el material empleado para llevar a cabo el proyecto. En primer lugar, se analizan las distintas bases datos que se recomiendan en la literatura y se justifica la escogida para el entrenamiento del modelo desarrollado en este proyecto. Una vez realizado lo anterior, se escoge el entorno de programación donde se va a desplegar el proceso de entrenamiento. Finalmente, en este apartado se describe el dispositivo donde se embebe el modelo programado, que en este caso se trata de una Raspberry Pi 4, justificando su elección y enumerando las ventajas e inconvenientes de la misma.

## 3.1 Estudio de Bases de Datos para *video captioning*

Puesto que el modelo que se ha implementado pertenece al ámbito del aprendizaje supervisado, es necesario seleccionar la base de datos adecuada que permita entrenar dicho modelo para distinguir múltiples actividades y entornos. Dado que dicho carácter generalista del modelo requiere una base de datos muy amplia, se ha recurrido a la literatura para buscar las bases de datos más completas con las que se pueden realizar este tipo de proyectos. A continuación, se muestra un listado de las bases de datos encontradas y un resumen de sus características más importantes.

En primer lugar, se muestra una recopilación de bases de datos que contienen distintas actividades de cocina:

**Max Planck Institute for Informatics (MPII)**

- 44 videos, cada uno 600 segundos de duración.
- Se muestra a 12 participantes preparando 14 tipos de platos.
- Se practican 65 tipos de actividades diferentes.
- Ejemplo de actividades: lavar manos, poner en un bol.
- La cámara y el escenario donde se cocina son fijos.

**You-Cook**

- 88 videos de extraídos de la plataforma digital Youtube.
- Para *machine learning* 49 vídeos entrenamiento 39 test.
- Cuenta con 6 estilos de cocina diferentes.
- 8 descripciones por video.
- La cámara y el escenario donde se cocina son diferentes en cada vídeo.

**TACoS (Textually Annotated Cooking Scenes)**

- Descripciones de video de alta calidad.
- 127 videos.
- Se practican 26 tipos de actividades diferentes.
- 20 descripciones diferentes para cada video.
- Es un subconjunto de MPII, comparten la característica de que la cámara y el escenario son fijos.

**TACoS MultiLevel**

- Subconjunto de la base de datos anterior (TACoS).
- Se generaron 3 niveles de descripción: detallada, corta y frase única.
- La descripción detallada como máximo cuenta con 15 frases por vídeo.
- La descripción corta cuenta con alrededor de 4 frases por vídeo.

**You-Cook II**

- 2000 videos con 89 recetas.
- Videos de Youtube.
- Variación en la posición de cámara, movimientos y escenario.
- Los videos se dividen en segmentos y se anota cada uno.
- 66% entrenamiento, 23% validación y 10% test.
- Los videos que se escogen para cada grupo se asignan de manera aleatoria.

De las bases de datos citadas anteriormente, serían de mayor utilidad las que tienen mayor diversidad de escenarios y ángulos de cámara. El mayor inconveniente que plantean es que todas sus actividades se limitan al entorno culinario, y para la aplicación del presente trabajo no sería suficiente. Es por ello por lo que se continúa evaluando distintas bases de datos, como las que se muestran a continuación, que contienen vídeos de distintas películas:

**MPII-MD (Movie Description Corpus)**

- Descripciones de audio de 94 películas de Hollywood.
- Películas subdivididas en clips de 4 segundos.
- Cada clip tiene al menos una frase asociada.
- También cuenta con subtítulos asociados al video.

**M-VAD**

- 48986 clips de 92 películas diferentes.
- Duración media del video de 6.2 segundos.
- Cada clip tiene al menos una frase asociada, algunos más.
- 38,949, 4,888 and 5,149 para entrenamiento, validación y test.

Aunque estos conjuntos de datos presentan mucha mayor variedad que en los anteriores, no presentan tantas descripciones por video como otros conjuntos que se recogerán en este documento. Las bases de datos que se citan a continuación contienen vídeos extraídos de redes sociales.

**VideoStory**

- 20000 videos de redes sociales
- Describen vídeos que son difíciles de sintetizar en una sola frase.
- Cada video cuenta con al menos un párrafo, con 4.67 frases en cada uno de media.
- Entrenamiento: 17908, Validación: 999 y test 1011.
- También se propone un grupo de test sin anotaciones publicadas de 1039 videos.

**ActivityNet Entities**

- Basado en Activity Net Captions dataset
- 14281 videos anotados
- 52000 segmentos con al menos un sintagma nominal para cada uno.

Por último, se muestran las bases de datos encontradas que tienen mayor diversidad de contenido.

**MSVD: Microsoft Video Description**

- 1970 Youtube Clips con anotaciones sin sonido
- Duración de 10 a 25 segundos
- Las anotaciones están en varios lenguajes
- La división típica del dataset suele ser 1200 entrenamiento, 100 validación y 670 test.

**MSR-VTT**

- 7180 videos subdivididos en 10000 clips.
- Videos agrupados en 20 categorías.
- 6513 entrenamiento, 497 validación y 2990 test.
- Cada video tiene 20 anotaciones, lo cual lo hace uno de los datasets más grandes.

**Charades**

- 9848 videos con actividades cotidianas
- 15 escenarios diferentes
- Se usan 46 objetos diferentes
- Duración de 30 segundos
- 7985 en entrenamiento y 1863 para validación/test

De entre estas últimas opciones, se ha escogido finalmente la base de datos **MSR-VTT**, ya que reúne las características necesarias para entrenar un modelo capaz de distinguir una gran diversidad de actividades y entornos.

## 3.2 Base de datos empleada: MSR-VTT

El conjunto de datos finalmente escogido para crear el modelo de descripción de escena fue adquirido por la compañía Microsoft y está pensado específicamente para modelos en los que se desee transformar video a texto. El contenido de esta base de datos es de 10.000 vídeos de la plataforma digital Youtube con una duración total de 41,2 horas y 200.000 descripciones de video en total, lo que permite no solo tener una gran cantidad de actividades, sino que se cuenta también con un gran número de descripciones.

Los 10.000 vídeos que se encuentran en este conjunto de datos se dividen como se muestra en la Tabla 1.

| Aplicación | Primer vídeo | Último video | Nº vídeos |
|---|---|---|---|
| **Entrenamiento** | Video0 | Video6512 | 6513 |
| **Validación** | Video6513 | Video7009 | 497 |
| **Test** | Video7010 | Video9999 | 2990 |

*Tabla 1 División del conjunto de datos MSR-VTT.*

La información de cada video se encuentra recogida en un fichero JSON que tiene la estructura que se muestra en la Tabla 2.

| Grupo | Subgrupo | Formato |
|---|---|---|
| **Info** | Year | Cadena de texto |
| | Version | Cadena de texto |
| | Description | Cadena de texto |
| | Contributor | Cadena de texto |
| | Data_created | Cadena de texto |
| **Videos** | Id | Entero |
| | Video_id | Cadena de texto |
| | Category | Entero |
| | Url | Cadena de texto |
| | Start time | Float |
| | End time | Float |
| | Split | Cadena de texto |
| **Sentences** | Sen_id | Entero |
| | Video_id | Cadena de texto |
| | Caption | Cadena de texto |

*Tabla 2. Estructura de datos del fichero JSON*

Los ficheros con los vídeos se encuentran dentro de las respectivas carpetas de entrenamiento, validación y test. A continuación, en la Ilustración 13, se muestra un ejemplo de los videos contenidos en el directorio de entrenamiento.

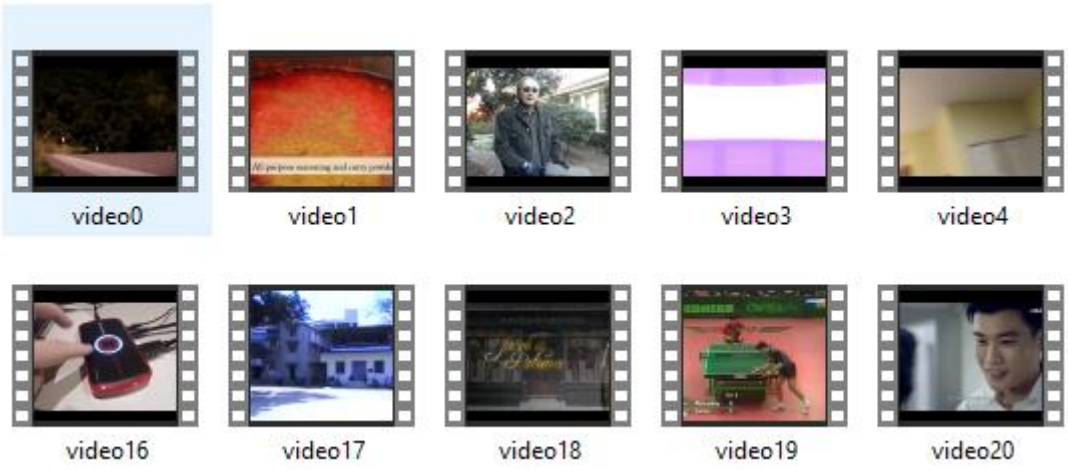



*Ilustración 13. Ejemplo directorio videos entrenamiento.*

Se ha recurrido a las herramientas que proporciona la librería OpenCV para descomponer cada vídeo en fotogramas y posteriormente asociar dichos fotogramas a su descripción de texto correspondiente. El ejemplo de un directorio que contiene la descomposición de un vídeo se encuentra en la Ilustración 14.

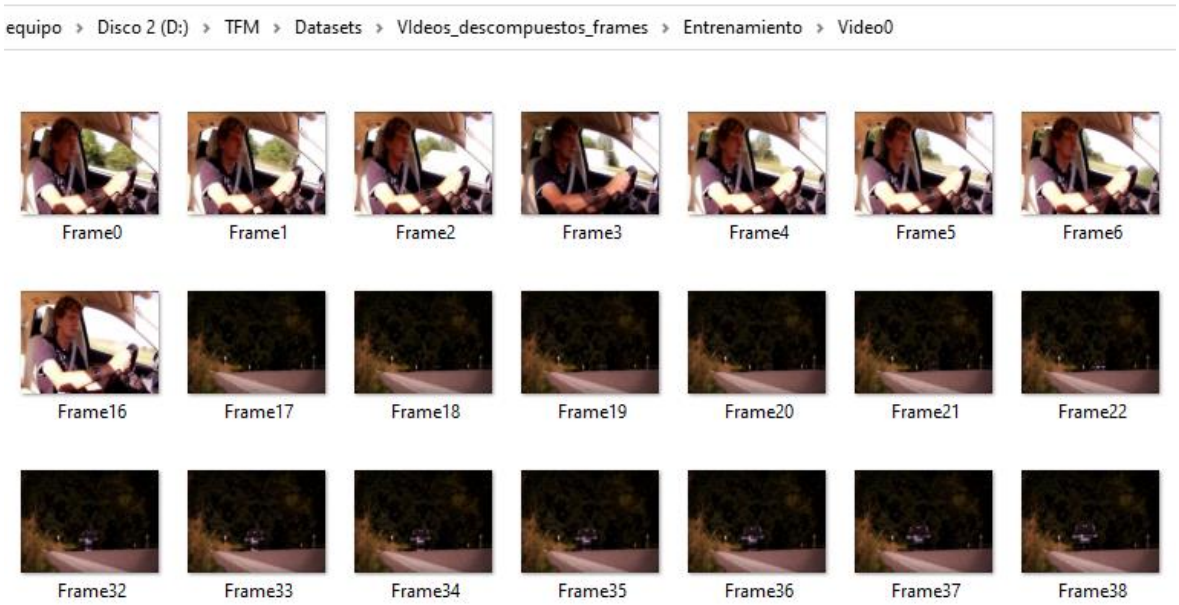



*Ilustración 14. Descomposición en fotogramas de vídeo0, tasa de 10 fotogramas por segundo.*

## 3.3 Google Colaboratory

En este trabajo se ha empleado el entorno de programación que ofrece Google, llamado Google Colaboratory. Este entorno es un entorno gratuito de Jupiter Notebook que no precisa de configuración previa y que ejecuta sus comandos en la nube. Su uso se basa en la creación de cuadernos, que son documentos que contienen código ejecutable y, si se desea, elementos de texto enriquecidos. Dichos cuadernos están compuestos por celdas, las cuales corresponden a bloques de ejecución. El logo representativo de Google Colaboratory se muestra en la Ilustración 15.

*Ilustración 15. Logo Google Colaboratory*

El modelo se ha implementado en un cuaderno de Google Colaboratory utilizando el lenguaje de programación Python. Gracias a que este entorno se encuentra en la nube, no es necesario instalar manualmente todas las librerías necesarias, así como sus dependencias puesto que esas a vienen configuradas en un entorno virtual. Esto proporciona al usuario una gran versatilidad a la hora de implementar programas que requieran integrar una gran cantidad de librerías diferentes.

De entre las distintas posibilidades que ofrece este servicio para ejecutar los cuadernos, se encuentran las siguientes:

- CPU (Central Process Unit)
- GPU (Graphics Grocessing Unit)
- TPU (*Tensor Processing Unit)*

En este proyecto se empleará una TPU. Este tipo de unidades son un hardware diseñado para resolver ciertas operaciones que se encuentran frecuentemente en procesos de aprendizaje automático, como la inferencia o entrenamiento de modelos. Las TPU se centran en operaciones vectoriales y sobre matrices. Su arquitectura las hace muy eficaces para operar con tensores.

Es importante emplear únicamente los recursos que se necesiten, los usuarios de este tipo de plataformas deben ser responsables y no asignar recursos elevados a tareas que no lo precisan. El buen funcionamiento de este tipo de infraestructuras en la nube depende mucho de ello.

## 3.4 Raspberry Pi 4

El dispositivo Raspberry Pi es un ordenador de placa reducida de bajo coste desarrollado por la *Raspberry Foundation* en Reino unido con el objetivo de iniciar a los alumnos de la escuela en la informática. En el año 2006, el primer diseño de la placa Raspberry Pi se basaba en el microcontrolador Atmel ATmega644, cuyo esquema se encuentra disponible públicamente para su descarga [48]. En 2009 se fundó la compañía Raspberry Pi en Caldecote, South Camdridgeshire, Reino Unido.

El software empleado es de código abierto, siendo el sistema operativo una versión adaptada de Debian, llamada Raspberry Pi OS. Todos los modelos cuentan con un procesador Broadcom, RAM, GPU, puertos USB, Ethernet, HDMI, conector para la cámara y 40 pines GPIO. La memoria es de tipo MicroSD, salvo la primera edición que era SD.

Los componentes principales de este dispositivo se detallan en la Tabla 3 y se muestran en la Ilustración 16.

| Numeración | Componente |
|---|---|
| **1** | 40 pines de propósito general |
| **2** | Entrada Ethernet |
| **3** | 2x USB 3.0 |
| **4** | 2x USB 2.0 |
| **5** | Salida Jack de audio |
| **6** | Cable plano para dispositivo de cámara |
| **7** | 2x Entrada MicroHDMI |
| **8** | Entrada USB tipo C (Alimentación) |
| **9** | Entrada MicroSD |
| **10** | Cable plano para dispositivo de pantalla |
| **11** | Bluetooth 5.0 – 2.4/5GHz Wireless |

*Tabla 3. Componentes de la Raspberrypi 4.*

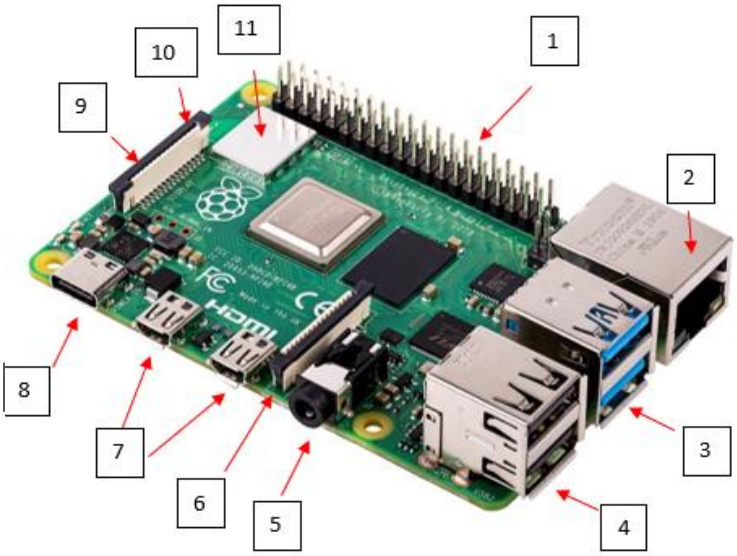



*Ilustración 16. Raspberry Pi 4* [49].

Las características técnicas de la Raspberrypi 4 se detallan en la Tabla 4.

| Especificaciones | |
|---|---|
| **Procesador** | ARM Cortex-A72 |
| **Frecuencia de reloj** | 1.5 Ghz |
| **GPU** | VideoCore VI (con soporte para OpenGL ES 3.x) |
| **Memoria** | 1 GB / 2 GB / 4 GB LPDDR4 SDRAM |
| **Conectividad** | Bluetooth 5.0, Wi-Fi 802.11ac, Gigabit Ethernet |

*Tabla 4. Especificaciones técnicas de la Raspberrypi 4*

Adicionalmente a la placa Raspberrypi, se han adquirido otros componentes que son necesarios para utilización. Debido a que la mayoría de los usuarios que adquieren estos equipos necesitan estos elementos adicionales, existen compañías que ofrecen packs para que el usuario no tenga que buscarlos de manera independiente. A continuación, se muestra en la Tabla 5 los componentes adquiridos:

| Componentes adicionales |
|---|
| **Tarjeta SanDisk 64GB Class 10 MicroSD con sistema Raspbian instalado (Ilustración 17)** |
| **3 disipadores de calor (Ilustración 18)** |
| **2 micro HDMI** |
| **5.1V 3A Tipo C con interruptor de ON / OFF** |
| **1 ventilador** |
| **1 lector de tarjetas** |
| **1 caja** |

*Tabla 5. Componentes adicionales adquiridos para la Raspberry Pi.*


*Ilustración 17. Tarjeta SanDisk 64GB MicroSD.*

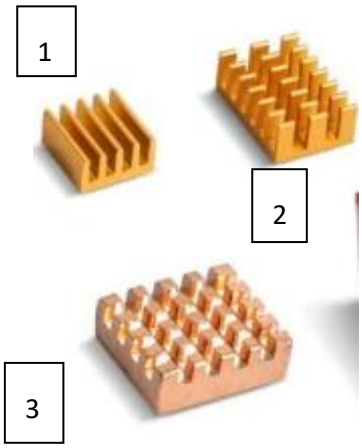



*Ilustración 18. Disipadores de calor*

Los disipadores de calor que se encuentran en la Ilustración 18 se deben ubicar manualmente en la placa. En la Ilustración 19 se muestra dónde deben ser ubicados. Es importante instalar estos disipadores ya que con algunos procesos de cómputo elevado la temperatura de la placa puede incrementarse a valores perjudiciales para la integridad de los circuitos electrónicos.

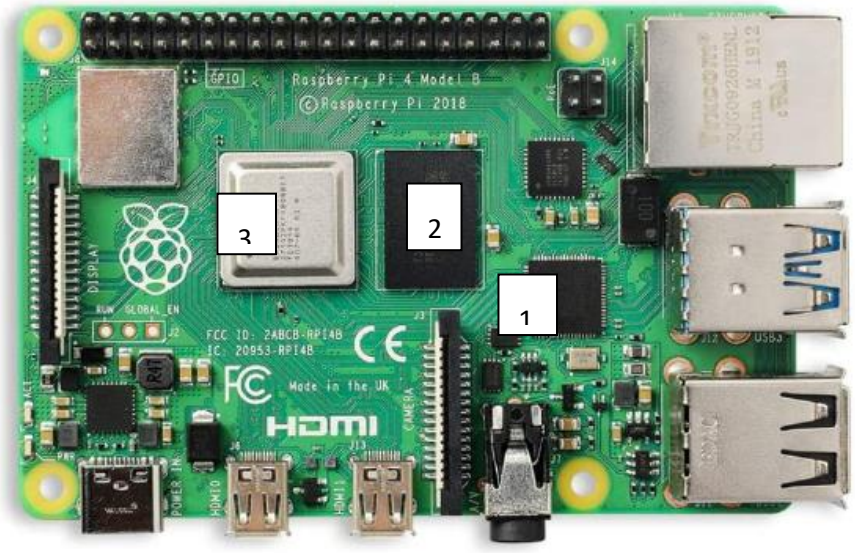



*Ilustración 19. Placa Raspberry Pi 4 con indicaciones de ubicación de los disipadores de calor.*

Los elementos restantes a los que hace referencia la Tabla 5 se muestran en la Tabla 6.

| | |
|---|---|
| *Cable MicroHDMI* |  |
| 5.1V 3A Tipo C con interruptor de ON / OFF. |  |
| Ventilador |  |
| Lector de tarjetas |  |
| *Caja para la placa Raspberry Pi 4.* |  |

*Tabla 6. Elementos restantes de la Raspberry Pi 4.*

El entrenamiento de modelos predictivos de aprendizaje profundo conlleva un elevado coste computacional. Por ello, este tipo de dispositivos, aunque cada vez cuentan con más memoria RAM y GPU, resultan ineficaces para estas tareas.

Por otra parte, aunque los procesos de entrenamiento no se puedan llevar a cabo en este tipo de dispositivos, si que es posible emplear un modelo previamente entrenado para llevar a cabo la fase de inferencia de nuevas muestras. Es por esta razón que en este proyecto se introduce el modelo entrenado en el dispositivo Raspberry pi 4.

Algunas de las ventajas que proporcionan este tipo de placas es que se puede simular el comportamiento del modelo desarrollado en un dispositivo portable. Los módulos complementarios de cámara permiten que el dispositivo, con tamaño muy reducido, sea capaz de adquirir la imagen, procesarla y proporcionar un resultado a la salida. Debido a la limitación de capacidad de cómputo, estos dispositivos ofrecen la posibilidad de poner a prueba la eficiencia del modelo.

Algunas de las desventajas a señalar es que debido a su poca memoria RAM y GPU, resultan demasiado lentos a la hora de procesar la información, lo cual es imprescindible en una aplicación real de este proyecto, ya que las descripciones de texto que se tienen que proporcionar deben ser prácticamente inmediatas. De igual manera, su poca capacidad de cómputo impide que los modelos predictivos puedan ser entrenados en ellos, solamente se pueden emplear los previamente entrenados. Otro aspecto a señalar es que le dispositivo empleado en este proyecto requiere de una fuente de alimentación, y por tanto, estar conectado a la red eléctrica. Aunque el tamaño de la Raspberry Pi es reducido, no lo es suficiente como para que se trate de un dispositivo cómodo, por lo que no sería una opción viable para una persona invidente.

## 3.5 Librerías empleadas

En este apartado se van a mencionar las librerías necesarias para poder ejecutar todas las funciones del programa. En este caso al emplearse Google Colaboratory, la mayoría de ellas ya se encuentran en la nube y basta con indicar en el cuaderno el nombre de la librería que se desea importar. En caso de emplear un entorno de programación en cualquier ordenador, sería necesario instalar dichas librerías. El listado es el siguiente:

- Librería ***TensorFlow***: librería imprescindible de código abierto desarrollada por Google pensada específicamente para el entrenamiento de redes neuronales. Dentro de esta librería se encuentra embebido el módulo Keras, que se explica a continuación. Esta librería fue diseñada por el equipo Google Brain y se liberó como código abierto en el año 2015.

- Librería **Keras:** se trata de una librería de redes neuronales artificiales de código abierto en lenguaje Python. Está pensada para aplicaciones en *Deep Learning*. El autor principal de esta librería es François Chollet, ingeniero de Google. En 2017 Tensorflow decidió dar soporta a Keras en su librería. La finalidad de Keras es proporcionar un entorno más intuitivo y sencillo para desarrollar modelos de *Deep Learning*.

- **NLTK:** se trata de un conjunto de librerías, conocido como kit de herramientas de lenguaje natural, que cuenta con programas para el procesamiento de lenguaje natural en el lenguaje de programación Python.

- Librería ***Json***: imprescindible para acceder a ficheros tipo JSON, en este proyecto es necesario ya que originalmente la información se encontraba en dicho formato.

- Librería ***Os:*** *permite ejecutar comandos como listar directorios para acceder a carpetas y leer los datos correspondientes****.***

- Librería ***Numpy***: esta librería contiene múltiples funciones que son imprescindibles para este proyecto. Proporciona mayor soporte para tratar vectores y matrices.

- Librería ***math:*** librería necesaria para realizar algunas operaciones matemáticas.

- Librería **pyttsx3**: librería imprescindible para generar el fichero de voz con la lectura de las descripciones generadas.

# 4 Descripción del modelo implementado

Esta sección del documento tiene como objetivo mostrar al lector el funcionamiento del modelo. Se muestra el esquema general del modelo y la estructura interna de los datos a medida que van atravesando las distintas etapas de la red neuronal.

## 4.1 Funcionamiento del modelo

El modelo diseñado consta de dos partes bien diferenciadas, un modelo de red neuronal convolucional seguido de un modelo de red neuronal recurrente. En primer lugar, se procesan las imágenes para ser introducidas en una red neuronal convolucional pre entrenada. La red neuronal convolucional empleada como extractor de características relevantes de la secuencia de imágenes ha sido la conocida como VGG16 (Visual Geometry Group). Dicha red ofrecerá a su salida un conjunto de características que el modelo deberá asociar a una determinada sentencia de texto.

Una vez extraídas las características de la imagen, se enviará dicha información a la siguiente parte del modelo, que corresponde a la red neuronal recurrente. La finalidad de este segundo modelo es captar la información temporal entre fotogramas del vídeo con el objetivo de predecir secuencialmente las palabras que conforman la frase que describe la imagen.

El esquema simplificado del funcionamiento de este modelo se muestra en la Ilustración 20. En los siguientes apartados de esta sección se describe con mayor detalle todo el funcionamiento del modelo mencionado.

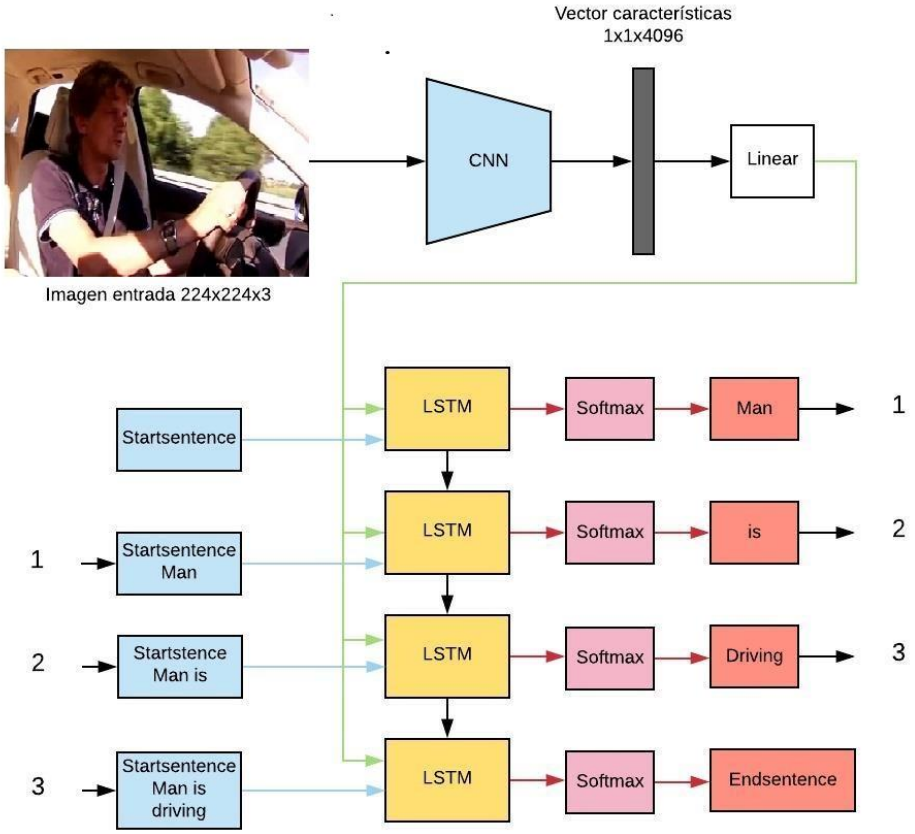



*Ilustración 20. Esquema del funcionamiento del modelo de redes neuronales diseñado.*

## 4.2 Red neuronal convolucional VGG16: Procesado de las imágenes

Aunque en este proyecto la fuente de información original se encuentre en forma de vídeos, a la hora de proporcionar información al modelo se deben enviar distintas imágenes con su descripción correspondiente. Como se ha mencionado en apartados anteriores, la finalidad de la red neuronal convolucional es procesar las imágenes de entrada y proporcionar un tensor a la salida que contiene las características extraídas de dicha imagen.

Para esta tarea se ha escogido la red neuronal convolucional VGG16, que es un modelo propuesto por K. Simonyan and A.Zisserman de la Universidad de Oxford [50]. Este modelo ha sido entrenado con la base de datos ImageNet durante semanas usando una gráfica NVIDIA Titan Black GPU's. ImageNet es un proyecto que contiene una gran cantidad de datos de imágenes naturalistas con sus respectivas anotaciones, es una de las más empleadas actualmente para tareas de visión artificial. El conocimiento extraído de toda la base de datos Imagenet se almacena en los pesos de los filtros que compone la red. Si se desea que la red VGG16 contenga dichos pesos con la información de la base de datos Imagenet, es necesario indicarlo cuando se carga el modelo. Existe una variante de esta arquitectura llamada

VGG19, pero es más compleja y resulta computacionalmente más pesada, razón por la que se emplea en este proyecto la VGG16.

La estructura que tiene la arquitectura de red VGG16 se muestra en la Ilustración 21, en ella se aprecia como la red transforma las imágenes de entrada para obtener un conjunto de características y clasificaciones. Aunque en la Ilustración 21 la salida sea un vector de 1x1000, en este proyecto se trabaja con las características sin clasificar, es decir, con la salida de la capa que proporciona un vector de 1x4096. Esto es debido a que lo que realmente interesa son las características extraídas por el modelo y su asociación con las descripciones de texto, y no la clasificación de la imagen en sí.

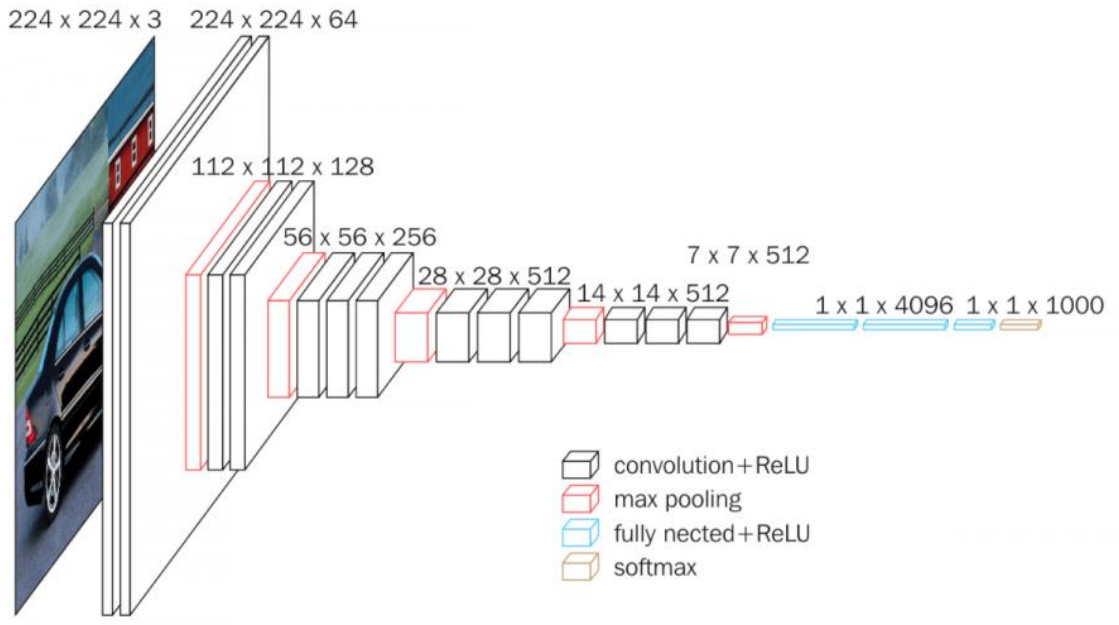



*Ilustración 21. Estructura de la red neuronal convolucional VGG16* [51].

Los distintos tipos de capas que se aprecian en la anterior ilustración se explican a continuación:

- **Capa convolucional**: las imágenes de entrada pasan a través de las capas de convolución donde cada filtro tiene un campo receptivo de 3x3 con un *stride* de 1. La resolución a la salida de estas capas no varía respecto a la de entrada.
- **Capa de *Pooling***: en esta capa de agrupación se aplica una ventana de tamaño máximo 2x2 con un stride de valor 2.
- **Capa *fully-connected***: las primeras capas fully-connected tienen 4096 canales cada una y la última cuenta con 1000, que corresponden a las categorías de las imágenes en la base de datos Imagenet.
- **Capa *softmax***: esta última etapa es la capa de activación final.

En el cuaderno de Google Colaboratory se muestra la configuración de las distintas capas de la red VGG16, como se puede apreciar en la Ilustración 22. Es importante señalar que el tamaño del modelo se encuentra alrededor de los 500 MB.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_1 (InputLayer) | [(None, 224, 224, 3)] | 0 |
| block1_conv1 (Conv2D) | (None, 224, 224, 64) | 1792 |
| block1_conv2 (Conv2D) | (None, 224, 224, 64) | 36928 |
| block1_pool (MaxPooling2D) | (None, 112, 112, 64) | 0 |
| block2_conv1 (Conv2D) | (None, 112, 112, 128) | 73856 |
| block2_conv2 (Conv2D) | (None, 112, 112, 128) | 147584 |
| block2_pool (MaxPooling2D) | (None, 56, 56, 128) | 0 |
| block3_conv1 (Conv2D) | (None, 56, 56, 256) | 295168 |
| block3_conv2 (Conv2D) | (None, 56, 56, 256) | 590080 |
| block3_conv3 (Conv2D) | (None, 56, 56, 256) | 590080 |
| block3_pool (MaxPooling2D) | (None, 28, 28, 256) | 0 |
| block4_conv1 (Conv2D) | (None, 28, 28, 512) | 1180160 |
| 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 |
| flatten (Flatten) | (None, 25088) | 0 |
| fc1 (Dense) | (None, 4096) | 102764544 |
| fc2 (Dense) | (None, 4096) | 16781312 |
| predictions (Dense) | (None, 1000) | 4097000 |

Total params: 138,357,544

*Ilustración 22. Modelo de red neuronal convolucional VGG16 cargado en Google Colaboratory.*

## 4.3 Preparación del texto

La finalidad de la preparación del texto es depurar la información con la que trabajará el modelo, con el fin de simplificar los cálculos lo máximo posible y evitar errores internos de funcionamiento. Para cada vídeo se cuenta con múltiples descripciones para cada instante. Debido a que la lectura del archivo JSON con la información es más lenta que trabajar con vectores y matrices generados en Python, se pre-procesa dicho archivo para generar una matriz que almacena pares vídeo/descripción para posteriormente leer de dicha matriz la información necesaria.

Es necesario realizar ciertas labores de depuración de texto antes de entrenar el modelo. Estas tareas principalmente son:

- Reducir todas las palabras a minúsculas
- Eliminar todos los signos de puntuación
- Eliminar todas las palabras que tienen una longitud de un carácter, como por ejemplo la preposición “a”
- Eliminar todos los números en las descripciones de texto
- Codificar todas las frases de manera que comiencen por la palabra “startsentence” y terminen por “endsentence”

Una vez se han llevado a cabo estas tareas, se puede comenzar a emplear las descripciones existentes para entrenar el modelo. En primer lugar, es importante analizar el vocabulario disponible en estas descripciones, tal y como se puede observar en la Ilustración 23.

```
El vocabulario tiene un tamaño de: 23109
La longitud máxima de descripción es: 71
```

*Ilustración 23. Tamaño del vocabulario y longitud de la frase más larga.*

La obtención de los parámetros mostrados en la Ilustración 23 es posible gracias al método *Tokenizer*, que permite crear un objeto que analiza todas las descripciones proporcionadas y calcular el tamaño de vocabulario, asociando para cada palabra un número entero.

## 4.4 Tratamiento de la información para el entrenamiento

Tanto la información recibida por parte de la etapa de procesado de imágenes y la de procesado de texto deben proporcionarse al algoritmo de una manera específica. Esto permite entrenar el modelo para llevar a cabo las predicciones deseadas.

Tal como se ha mostrado en la Ilustración 20, en la Sección 4.1, el modelo desarrollado recibe como entrada las características extraídas de una imagen y predice palabra por palabra la descripción de esta. Para la etapa de entrenamiento, las descripciones de texto deben ser introducidas progresivamente palabra por palabra. En la Ilustración 24 se muestra un ejemplo de la forma que tiene la información que procesará el modelo:


*Ilustración 24. Imagen de ejemplo, extraída del video 20 del conjunto de datos MSR-VTT.*

La descripción asociada a la Ilustración 24 es "a child being presented a bowl". Después de la etapa de depuración la descripción habrá quedado como "startsentence child being presented bowl endsequence".

La longitud de la descripción es de seis palabras, por lo que al modelo se le introduce progresivamente la descripción de texto a la que se le añade cada vez una palabra, comenzando por "startsentence". La Tabla 7 muestra el aspecto que tiene la información en este proceso.

| X (Imagen 1x4096) | X2 (secuencia de texto) | Y (palabra a predecir) |
|---|---|---|
| **Imagen** | Startsentence | child |
| **Imagen** | Startsentence, child | being |
| **Imagen** | Startsentence, child, being | presented |
| **Imagen** | Startsentence, child, being, presented | bowl |
| **Imagen** | Startsentence, child, being, presented, bowl | endsentence |

*Tabla 7. Ejemplo de información de entrenamiento del modelo diseñado.*

Aunque en la Tabla 7 muestra que las descripciones tienen formato de texto, el algoritmo de aprendizaje automático no trabaja de esta manera. Como se ha comentado en el apartado anterior, existe un objeto denominado "Tokenizer" que se encarga de analizar el vocabulario presente en todas las descripciones dadas y asigna un número entero a cada palabra. Esto permite codificar cada frase

en una sucesión de números enteros, lo que permite al modelo trabajar prediciendo valores numéricos. Estos valores una vez obtenidos se decodifican buscando la palabra asociada a cada uno para conformar la descripción de salida del modelo.

En este caso, el vocabulario está compuesto por 23109 palabras y la frase de mayor longitud tiene 71 palabras. Por tanto, la información vendrá proporcionada como una imagen en forma de tensor de 1x4096, una secuencia de texto progresiva codificada en números enteros y una palabra a predecir de salida codificada en *one-hot encoding*. Todo esto se muestra en la Tabla 8 y Tabla 9.

| Palabra | Codificación |
|---|---|
| **Starsequence** | 1 |
| **Child** | 163 |
| **Being** | 39 |
| **Presented** | 1160 |
| **Bowl** | 151 |
| **Endsequence** | 2 |

*Tabla 8. Codificación de las palabras contenidas en el ejemplo de tratamiento de la información.*

| X (Imagen 1x4096) | X2 (secuencia de enteros) | Y (entero a predecir, one-hot) |
|---|---|---|
| **Imagen** | [1] | [163] |
| **Imagen** | [1], [163] | [39] |
| **Imagen** | [1], [163], [39] | [1160] |
| **Imagen** | [1], [163], [39], [1160] | [151] |
| **Imagen** | [1], [163], [39], [1160], [151] | [2] |

*Tabla 9. Ejemplo de codificación de la información de entrenamiento del modelo diseñado*

La codificación asigna los primeros valores enteros a las palabras más frecuentes. Puesto que todas van codificadas con una palabra de inicio y fin, es lógico que los primeros valores sean asignados a dichas palabras. El modelo predice palabras hasta que se obtiene la palabra de fin, momento en el cual muestra la descripción de texto completa.

## 4.5 Carga progresiva de los datos

La carga progresiva de datos es imprescindible para este proyecto. Debido a la gran cantidad de videos y fotogramas en este trabajo, no es posible manejar toda la información necesaria de manera simultánea sin exceder la capacidad de memoria RAM del entorno. Esto sucede incluso empleando los recursos de Google Colaboratory.

Para manejar toda la información sin saturar la memoria RAM, es necesario definir un objeto conocido como “Datagenerator” que recorra toda la información y vaya cargándola progresivamente. Esto permite que se almacene temporalmente únicamente una cantidad limitada de información, y se vaya accediendo a esta de manera ordenada. Para indicarle al objeto en qué momento debe de parar, es importante indicarle al método de entrenamiento, el número de pasos por época.

# 5 Resultados

En esta sección del documento se muestran las métricas empleadas para la evaluación de los resultados, ensayos realizados, problemas encontrados durante el desarrollo del proyecto y su implementación en el dispositivo portable Raspberry Pi 4.

## 5.1 Métricas de evaluación

De entre las distintas métricas empleadas en aprendizaje automático para evaluar las traducciones y descripciones automáticas, destacan BLEU, METEOR y ROUGE-L, por lo que estas serán las que se emplearán en este proyecto para evaluar la calidad del modelo entrenado. Todas las métricas presentadas ofrecen resultados entre 0 y 1, siendo 0 el peor resultado y 1 el mejor posible.

BLEU es una sigla en inglés cuyo significado es "Bilingual Evaluation Understudy". La calidad de la descripción generada depende de lo similar que sea a una de referencia, la cual es supuestamente correcta. Se puede emplear para su cálculo más de una frase de referencia, siempre eligiendo el valor máximo obtenido en esta métrica. De esta manera, se dota a la evaluación de mayor robustez.

BLEU se utiliza para calificar la similitud entre frases y para ello calcula la precisión entre ngramas de ambas frases. Ngrama es el término específico que se emplea para designar a una palabra que forma parte de una oración. Dicha precisión entrene ngramas viene dada por la siguiente expresión:

$$Precisión = \frac{ngramas\ comunes}{ngramas\ candidata}$$

*Ecuación 6. Cálculo de la precisión en BLEU.*

Los ngramas comunes son los que se encuentran tanto en la frase candidata, como en la frase de referencia. A continuación, se muestra un ejemplo de cálculo de esta precisión:

- **Frase de referencia**: Un hombre viaja en un avión
- **Frase candidata 1**: Un viajero vuela en el avión
- **Frase candidata 2**: El hombre viaja en un avión

Respecto a la frase candidata 1, se identifican en la Tabla 10 los ngramas comunes y en la Tabla 11 los distintos.

| Ngramas comunes candidata 1 |
|---|
| Un |
| En |
| Avión |

*Tabla 10. Ngramas comunes para frase candidata 1 de ejemplo.*

| Ngramas diferentes candidata 1 |
|---|
| Viajero |
| Vuela |
| El |

*Tabla 11. Ngramas diferentes para frase candidata 1 de ejemplo.*

Siendo que el número de ngramas que componen la frase es de 6, y que hay 3 de los cuales son comunes, la precisión de la candidata 1 resulta de:

$$P_1 = \frac{3}{6}$$

Realizando el mismo procedimiento para la frase candidata 2, se identifican los ngramas comunes y distintos en la Tabla 12 y Tabla 13 respectivamente.

| Ngramas comunes candidata 2 |
|---|
| Hombre |
| Viaja |
| En |
| Un |
| Avión |

*Tabla 12. Ngramas comunes para frase candidata 2 de ejemplo.*

| Ngramas diferentes candidata 2 |
| --- |
| **El** |

*Tabla 13. Ngramas diferentes para frase candidata 1 de ejemplo.*

El número de ngramas comunes es de 5 y el total de 6. En este caso, el cálculo de la precisión resulta:

$$P_2 = \frac{5}{6}$$

Se confirma que la frase candidata 2 tiene mayor calidad que la primera, lo cual podía deducirse de manera intuitiva por el lector. Ahora bien, esta métrica tiene como principal problema que puede arrojar resultados de alta precisión, aunque la traducción no sea correcta, lo cual se ilustra con el siguiente ejemplo:

- **Frase de referencia**: Un hombre viaja en un avión
- **Frase candidata 3**: Un un un un un un

En este caso, solamente se cuenta con un ngrama común, pero éste se repite 6 veces. Lo que se cuenta al tener en cuenta los ngramas comunes no son las unidades sino las apariciones, por lo que al aparecer en la frase canditata 6 veces un ngrama común, la precisión resultaría lo siguiente:

$$P_3 = \frac{6}{6} = 1$$

Esta frase candidata ofrece el resultado máximo de precisión, aunque no cabe duda de que es errónea. Para solventar este problema se utiliza la precisión modificada que consiste en tener en cuenta el número de veces que aparece un ngrama en la frase de referencia. Dado que en este caso solo aparece el ngrama “Un” dos veces en la frase de referencia, la precisión modificada queda:

$$P_{3\,modificada} = \frac{2}{6}$$

Siendo este resultado más coherente que el anterior, quedando esta opción como la peor candidata.

Además existe una penalización por brevedad, ya que una frase compuesta por una palabra podría obtener una puntuación de “1” en la precisión. Esta penalización viene dada por la siguiente expresión:

$$PB = \begin{cases} 1 & c > r \\ e^{1-\frac{r}{c}} & c \leq r \end{cases}$$

*Ecuación 7. Expresión del coeficiente de penalización por brevedad para el cálculo BLEU.*

Donde $c$ es la longitud de la frase candidata y $r$ la longitud de la frase de referencia.

Finalmente, el cálculo de BLEU viene dado por la expresión que se muestra a continuación:

$$BLEU = PB * e^{(\sum_{n=1}^{n} w_n * log P_n)}$$

*Ecuación 8. Expresión del cálculo de BLEU.*

Respecto a la nomenclatura de la Ecuación 8, cada elemento significa lo siguiente:

- PB: coeficiente de penalización por brevedad
- $W_n$: peso de cada ngrama, la suma de todos los pesos debe ser igual a 1
- $P_n$: precisión entre ngramas

Esta métrica se considera demasiado rígida, ya que no contempla sinónimos ni variaciones léxicas.

La siguiente métrica que se va a emplear recibe el nombre de METEOR. Se trata de una métrica empleada para la evaluación de traducciones automáticas. Al igual que BLEU, esta métrica también recibe su nombre por una sigla cuyo significado es "Metric for Evaluation for Translation with Explicit Ordering". Esta métrica está basada en la media armónica de precisión y recuperación del ngrama. Lo que la hace particularmente útil frente a otras métricas como BLEU es que tiene en cuenta las coincidencias como las derivaciones o sinónimos, aportando mayor flexibilidad a la evaluación de las traducciones generadas.

El cálculo de METEOR viene dado por la expresión que se muestra a continuación, en la Ecuación 9.

$$\text{METEOR} = \text{F}_{\text{mean}} * (1 - p)$$

*Ecuación 9. Expresión del cálculo de la métrica METEOR.*

La nomenclatura de los términos que aparecen en la Ecuación 9 es la siguiente:

- $F_{mean}$: media armónica que combina los cálculos de precisión y recuperación del ngrama, los cuales se explicarán a continuación.
- $p$: cálculo de la penalización, cuyo valor máximo es de 0.5

El término de la media armónica en la Ecuación 9 se calcula de la siguiente manera:

$$F_{mean} = \frac{10PR}{R + 9P}$$

*Ecuación 10. Expresión del cálculo de la media armónica de la precisión y la recuperación del ngrama.*

Los términos precisión y recuperación del ngrama presentes en la Ecuación 10 se calculan como se muestra a continuación:

$$P = \frac{m}{w_t}$$

*Ecuación 11. Precisión del ngrama*

$$R = \frac{m}{w_r}$$

*Ecuación 12. Recuperación del ngrama.*

Los términos que aparecen en la Ecuación 11 y Ecuación 12 tienen el siguiente significado:

- $m$: número de ngramas de coincidencia entre la traducción de referencia y la candidata.
- $w_t$: número de ngramas totales de la traducción candidata.
- $w_r$: número de ngramas totales de la traducción de referencia.

Por otra parte, el cálculo de la penalización viene dado por la siguiente expresión:

$$p = 0.5\left(\frac{c}{u_m}\right)^3$$

- $c$: número de trozos
- $u_m$: número de unigramas mapeados

Respecto a la última métrica empleada, ROUGE, esta significa "Recall-Oriented Understudy for Gisting evaluation", y se trata de un conjunto de métricas pensadas para la evaluación de traducciones generadas en base a la comparación con una referencia. ROUGE-L es una métrica que se encuentra dentro de este conjunto y trata de evaluar las traducciones en base a la subsecuencia común más larga (*LCS* en inglés). Este método considera la similitud de estructura como oración de forma natural e identifica los ngramas de secuencia más largos concomitantes.

## 5.2 Análisis de resultados obtenidos

Una vez definidas las métricas que se van a emplear para evaluar la eficacia de los modelos implementados y solucionadas las dificultades encontradas, se han realizado cinco ensayos distintos. Los ensayos realizados son los que se detallan a continuación:

- **Ensayo 1**: entrenamiento del modelo sin dropout durante 100 épocas
- **Ensayo 2**: entrenamiento del modelo con un valor de dropout del 10% durante 100 épocas
- **Ensayo 3**: entrenamiento del modelo con un valor de dropout del 20% durante 100 épocas
- **Ensayo 4**: entrenamiento del modelo con un valor de dropout de 40% durante 100 épocas
- **Ensayo 5**: entrenamiento del modelo con un valor de dropout de 50% durante 100 épocas.

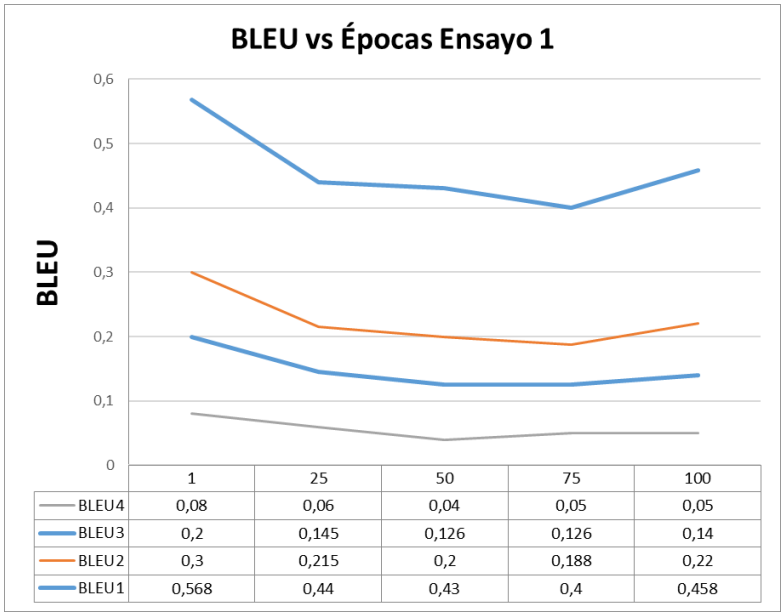
Los resultados obtenidos tras la evaluación de cada una de las tres métricas en los ensayos realizados se resumen en la Tabla 14.

| Ensayo | Dropout | Épocas | BLEU1 | BLEU2 | BLEU3 | BLEU4 | ROUGE-L | METEOR |
|---|---|---|---|---|---|---|---|---|
| Primer ensayo | No | 1 | 0,568 | 0,3 | 0,2 | **0,08** | 0,392 | 0,156 |
| | No | 25 | 0,44 | 0,215 | 0,145 | 0,06 | 0,298 | 0,174 |
| | No | 50 | 0,43 | 0,2 | 0,126 | 0,04 | 0,23 | 0,158 |
| | No | 75 | 0,4 | 0,188 | 0,126 | 0,05 | 0,243 | 0,181 |
| | No | 100 | 0,458 | 0,22 | 0,14 | 0,05 | 0,32 | 0,171 |
| Segundo ensayo | 0,1 | 1 | 0,558 | **0,299** | **0,199** | **0,08** | 0,345 | 0,149 |
| | 0,1 | 25 | 0,466 | 0,23 | 0,156 | 0,069 | 0,307 | 0,147 |
| | 0,1 | 50 | 0,461 | 0,225 | 0,145 | 0,05 | 0,302 | 0,124 |
| | 0,1 | 75 | 0,425 | 0,203 | 0,126 | 0,04 | 0,257 | 0,152 |
| | 0,1 | 100 | 0,437 | 0,209 | 0,133 | 0,05 | 0,254 | 0,179 |
| Tercer ensayo | 0,2 | 1 | 0,565 | 0,297 | 0,191 | 0,07 | 0,339 | 0,172 |
| | 0,2 | 25 | 0,468 | 0,245 | 0,167 | 0,07 | 0,269 | 0,122 |
| | 0,2 | 50 | 0,481 | 0,239 | 0,16 | 0,07 | 0,308 | 0,12 |
| | 0,2 | 75 | 0,493 | 0,256 | 0,174 | 0,07 | 0,296 | 0,125 |
| | 0,2 | 100 | 0,523 | 0,272 | 0,178 | 0,08 | 0,333 | 0,163 |
| Cuarto ensayo | 0,4 | 1 | 0,527 | 0,279 | 0,183 | **0,08** | 0,372 | 0,15 |
| | 0,4 | 25 | 0,527 | 0,278 | 0,183 | **0,08** | 0,367 | 0,147 |
| | 0,4 | 50 | 0,486 | 0,237 | 0,157 | 0,06 | 0,357 | 0,172 |
| | 0,4 | 75 | 0,507 | 0,246 | 0,159 | 0,06 | 0,365 | 0,21 |
| | 0,4 | 100 | 0,453 | 0,224 | 0,141 | 0,05 | 0,3 | 0,2 |
| Quinto ensayo | 0,5 | 1 | 0,55 | 0,287 | 0,19 | **0,08** | 0,35 | 0,187 |
| | 0,5 | 25 | 0,514 | 0,276 | 0,187 | **0,08** | 0,352 | 0,152 |
| | 0,5 | 50 | 0,462 | 0,221 | 0,155 | 0,06 | 0,28 | 0,162 |
| | 0,5 | 75 | 0,5 | 0,235 | 0,151 | 0,06 | 0,4 | 0,158 |
| | 0,5 | 100 | **0,572** | 0,257 | 0,166 | 0,07 | **0,41** | **0,188** |

*Tabla 14. Resultados obtenidos del cálculo de las métricas de evaluación para cada ensayo realizado.*

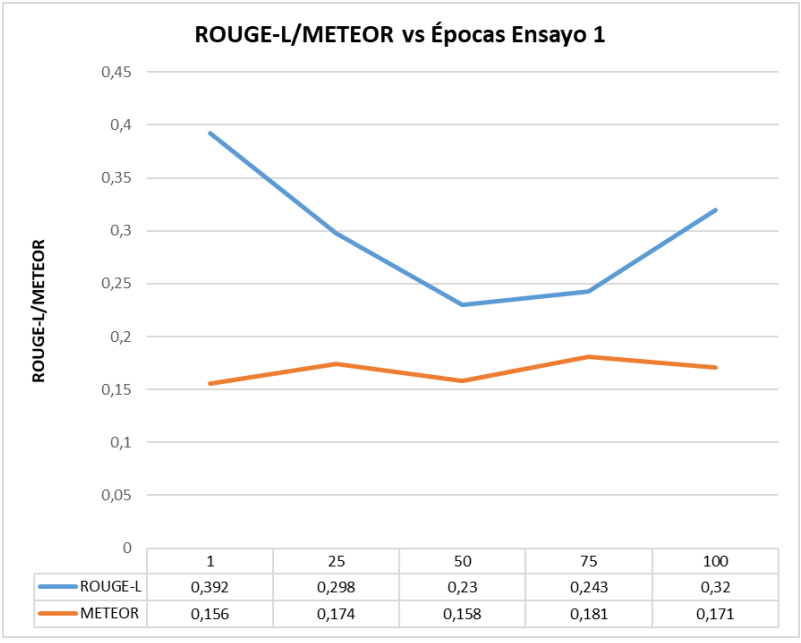
En la Tabla 14 se han destacado los valores máximos para cada métrica. Puesto que el modelo entrenado durante 100 épocas y con dropout del 50% es el que alcanza el valor máximo en las tres métricas, este modelo será el empleado para ser embebido en la Raspberry Pi 4.

A continuación, se van a analizar los datos obtenidos para cada ensayo, comenzando por el primero, en la Ilustración 25 e Ilustración 26 se observa como varían las métricas en función del número de épocas.




| | 1 | 25 | 50 | 75 | 100 |
|---|---|---|---|---|---|
| BLEU4 | 0,08 | 0,06 | 0,04 | 0,05 | 0,05 |
| BLEU3 | 0,2 | 0,145 | 0,126 | 0,126 | 0,14 |
| BLEU2 | 0,3 | 0,215 | 0,2 | 0,188 | 0,22 |
| BLEU1 | 0,568 | 0,44 | 0,43 | 0,4 | 0,458 |

*Ilustración 25. Evolución de la métrica BLEU para el ensayo 1 en función del número de épocas.*



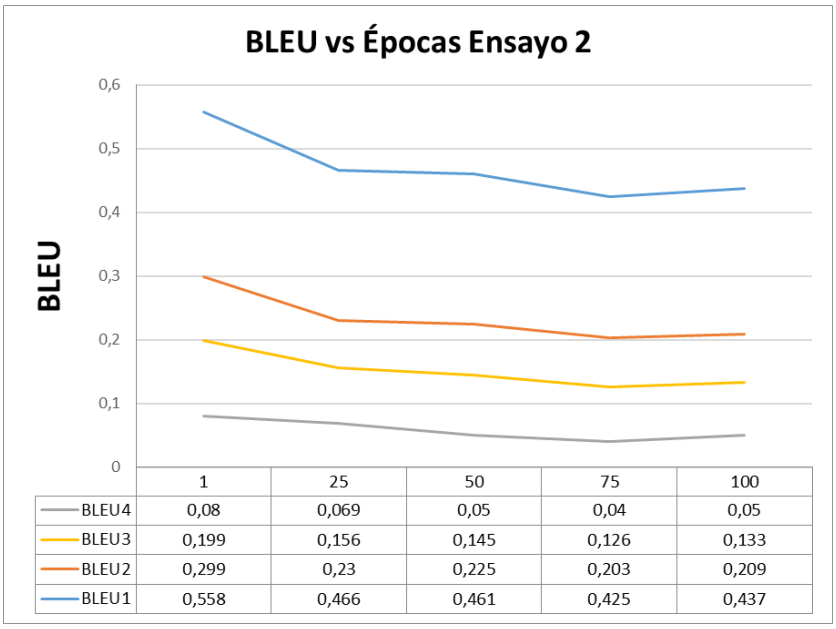

| | 1 | 25 | 50 | 75 | 100 |
|---|---|---|---|---|---|
| ROUGE-L | 0,392 | 0,298 | 0,23 | 0,243 | 0,32 |
| METEOR | 0,156 | 0,174 | 0,158 | 0,181 | 0,171 |

*Ilustración 26. Evolución de las métricas ROUGE-L y METEOR para el ensayo 1 en función del número de épocas.*

Tanto en la Ilustración 25 como en la Ilustración 26 se observa que en primer lugar los valores de las métricas tienen su valor más elevado y después decrecen progresivamente, observando un pequeño cambio de tendencia en las últimas épocas donde parece aumentar su valor. Una posible explicación a este fenómeno es lo que se conoce como sobreajuste (*overfitting* en inglés), pues el modelo implementado en primer lugar sí que es capaz de realizar predicciones de mejor calidad sobre el conjunto de datos de test pero a medida que transcurren las épocas, el modelo se va ajustando a los datos de entrenamiento y pierde la capacidad de generalizar para conjuntos de datos no observados, como son los datos de test.

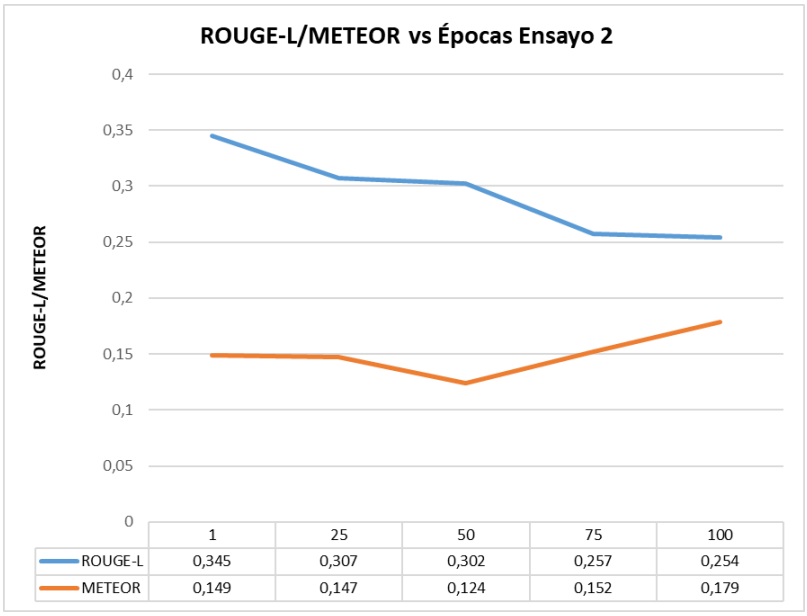
Aunque en las últimas épocas se aprecia un leve incremento de rendimiento, el elevado coste computacional que comportaría explorar un mayor número de épocas deja fuera del alcance de este proyecto dicha evaluación. Así pues, se va a estudiar en los siguientes ensayos la introducción progresiva de mayores valores de dropout para observar su efecto sobre el rendimiento y el sobreajuste del modelo.

En primer lugar, se introduce un valor de dropout del 10% y se evalúa la evolución de las métricas obtenidas en la Ilustración 27 e Ilustración 28.




| | 1 | 25 | 50 | 75 | 100 |
|---|---|---|---|---|---|
| BLEU4 | 0,08 | 0,069 | 0,05 | 0,04 | 0,05 |
| BLEU3 | 0,199 | 0,156 | 0,145 | 0,126 | 0,133 |
| BLEU2 | 0,299 | 0,23 | 0,225 | 0,203 | 0,209 |
| BLEU1 | 0,558 | 0,466 | 0,461 | 0,425 | 0,437 |

*Ilustración 27. Evolución de la métrica BLEU para el ensayo 2 en función del número de épocas.*



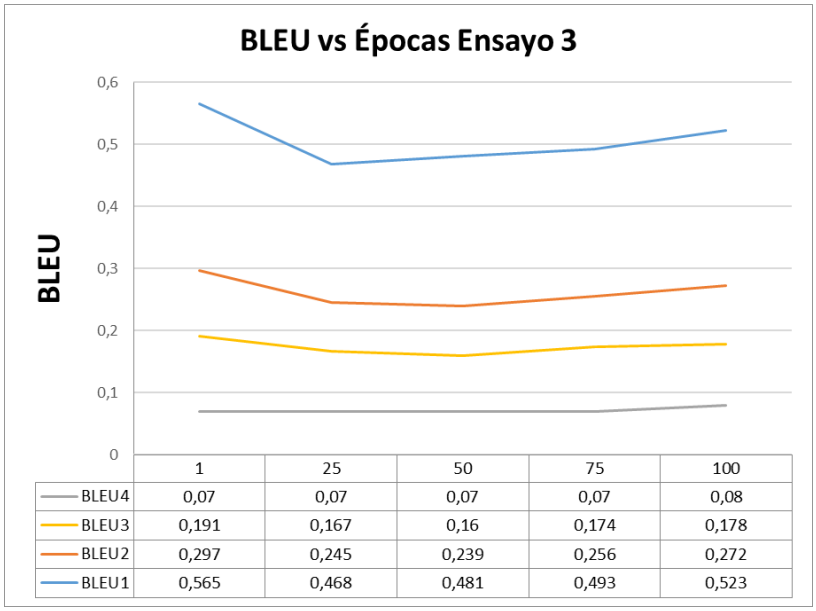

| | 1 | 25 | 50 | 75 | 100 |
|---|---|---|---|---|---|
| ROUGE-L | 0,345 | 0,307 | 0,302 | 0,257 | 0,254 |
| METEOR | 0,149 | 0,147 | 0,124 | 0,152 | 0,179 |

*Ilustración 28. Evolución de las métricas ROUGE-L y METEOR para el ensayo 2 en función del número de épocas.*

Respecto a la métrica BLEU, como se aprecia en la Ilustración 27, no se un incremento de rendimiento significativo respecto al primer ensayo, aunque se aprecia que el cambio de tendencia final parece menos acusado, estabilizándose el valor de la métrica más que en el ensayo anterior. Esto puede ser debido a que el valor de dropout introducido no tiene un valor elevado y por tanto, no afecta de manera significativa al rendimiento del modelo entrenado.

En relación con la métrica METEOR y ROUGE-L, sí que se aprecia un cambio de tendencia respecto al primer ensayo. Mientras que en este caso la métrica ROUGE-L tiene una tendencia descendente y aparentemente convergente para valores de épocas altos, la métrica METEOR no parece tener una tendencia clara en función del número de épocas, presentando una tendencia ascendente para valores de épocas elevados.

A continuación, se muestran los resultados obtenidos en el ensayo 3 en la Ilustración 29 e Ilustración 30, para el cual se ha introducido un valor de dropout del 20%.




| | 1 | 25 | 50 | 75 | 100 |
|---|---|---|---|---|---|
| BLEU4 | 0,07 | 0,07 | 0,07 | 0,07 | 0,08 |
| BLEU3 | 0,191 | 0,167 | 0,16 | 0,174 | 0,178 |
| BLEU2 | 0,297 | 0,245 | 0,239 | 0,256 | 0,272 |
| BLEU1 | 0,565 | 0,468 | 0,481 | 0,493 | 0,523 |

*Ilustración 29. Evolución de la métrica BLEU para el ensayo 3 en función del número de épocas.*

Para la métrica BLEU, como se puede observar en la Ilustración 29, se produce una primera tendencia descendente en las primeras épocas, pero después se produce un punto de inflexión que marca el comienzo de una tendencia ascendente hasta el final del rango de épocas evaluadas. Los resultados de estas métricas son mayores que en los anteriores ensayos para un mismo número de épocas, luego se puede determinar que, para una capacidad de cómputo limitada, la introducción de dropout puede proporcionar mejores resultados para el mismo número de épocas.

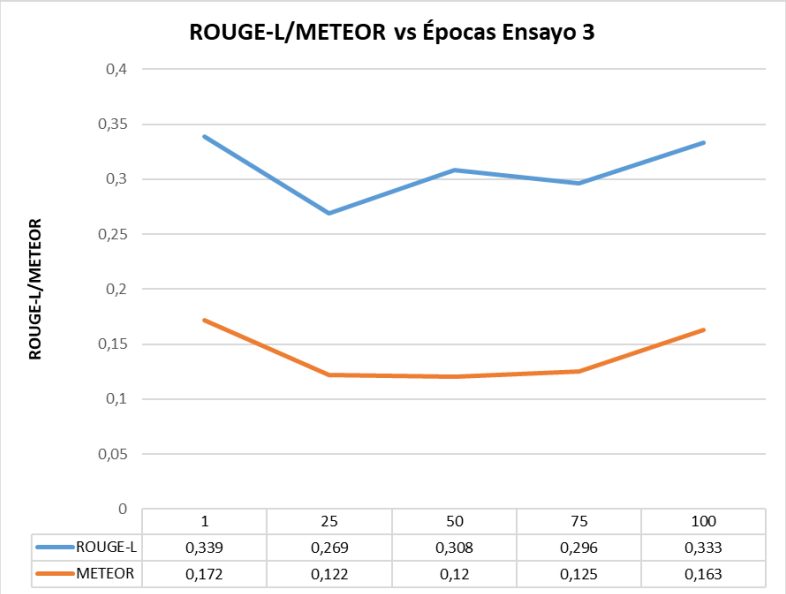



*Ilustración 30. Evolución de las métricas ROUGE-L y METEOR para el ensayo 3 en función del número de épocas.*

Adicionalmente, se aprecia tanto en la métrica ROUGE-L como la METEOR una primera tendencia descendente que cambia para crecer progresivamente. Los resultados obtenidos para este ensayo indican que, según estas dos métricas, los modelos obtenidos en este ensayo son de peor calidad que los del primero. Se recuerda que la métrica BLEU es bastante rígida a la hora de evaluar, por lo que es posible que se de esta disparidad a la hora de evaluar los modelos obtenidos. Es importante evaluar el conjunto de todas las métricas para garantizar que se escoge el modelo óptimo.

En el siguiente ensayo se introduce el doble de dropout que en el anterior con un valor del 40%. Los resultados obtenidos se muestran en la Ilustración 31 e Ilustración 32.

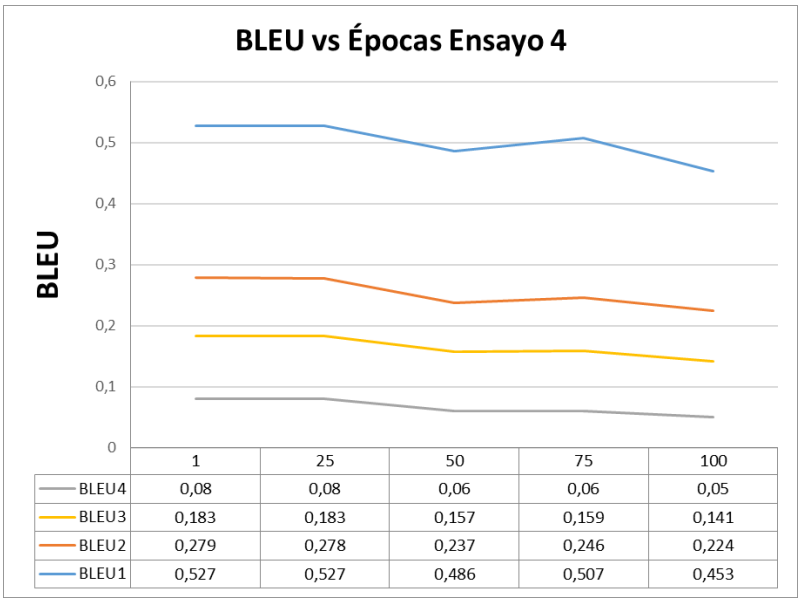



*Ilustración 31. Evolución de la métrica BLEU para el ensayo 4 en función del número de épocas.*

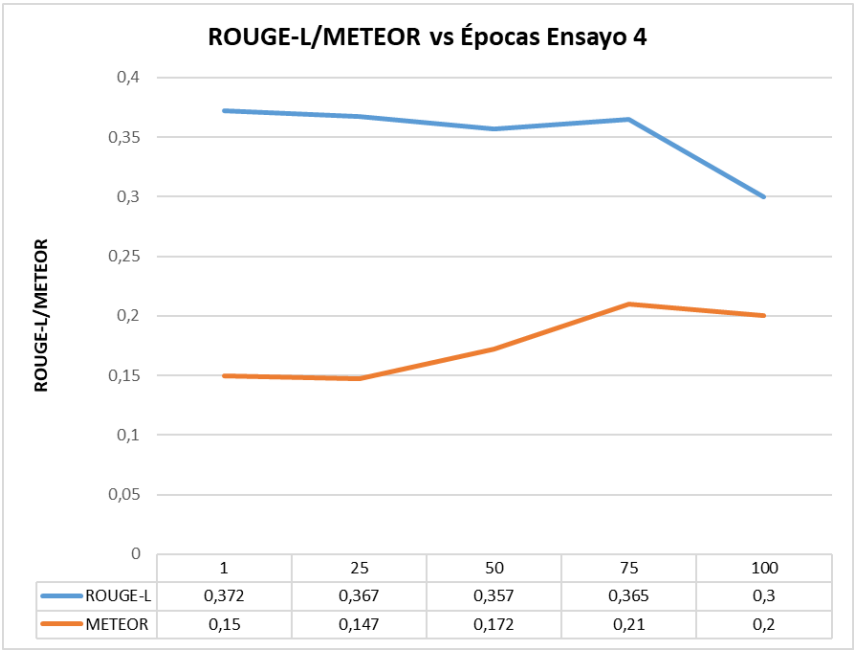
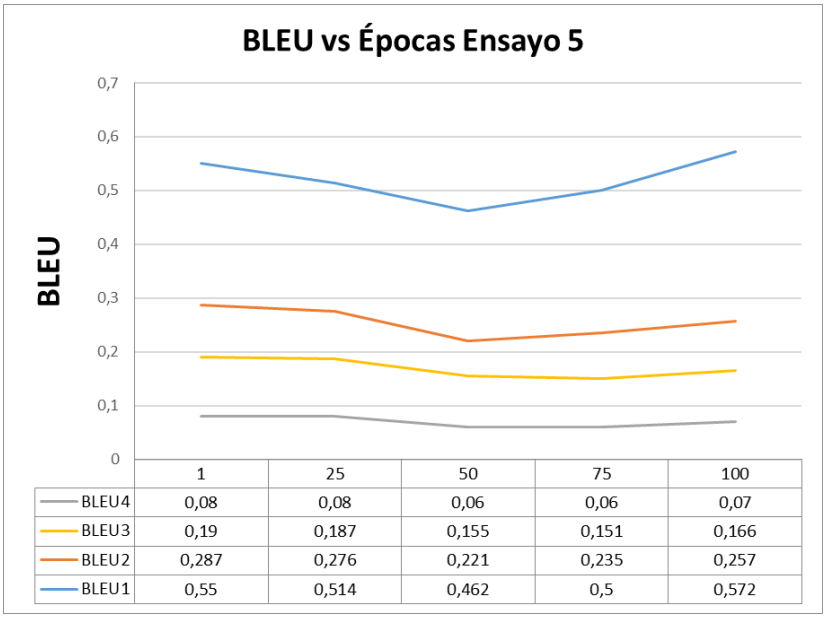



*Ilustración 32. Evolución de las métricas ROUGE-L y METEOR para el ensayo 4 en función del número de épocas.*

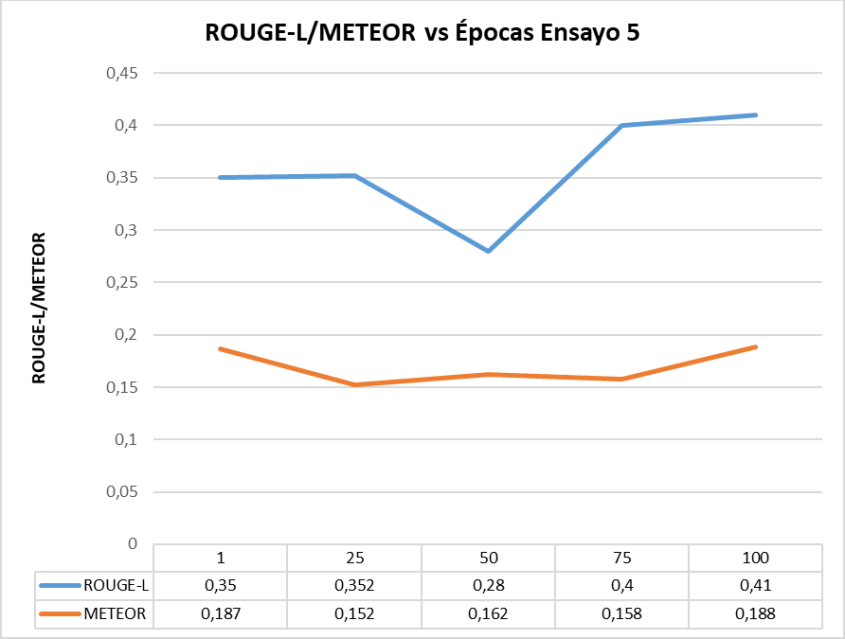
En la Ilustración 31 se aprecia que la métrica BLEU adquiere valores más estables y a medida que avanza el número de épocas disminuye el rendimiento en el conjunto de datos de test. Se observa que, a partir de 75 épocas, el efecto del sobreajuste empieza a ser más notable y por lo tanto, disminuye en mayor medida el rendimiento del modelo. Aunque la métrica BLEU ha proporcionado mejor resultados en otros ensayos, en este caso se aprecia una mayor robustez ya que hasta cierto número de épocas el modelo parece no sufrir el efecto del sobreajuste. En el caso de las métricas ROUGE-L y METEOR, los resultados obtenidos son mejores y más robustos que para los ensayos anteriores, aunque a partir de las 75 épocas se produce un elevado descenso de rendimiento por sobreajuste.

Por último, se muestran los resultados obtenidos con un valor de dropout del 50% en la Ilustración 33 e Ilustración 34.




| | 1 | 25 | 50 | 75 | 100 |
|---|---|---|---|---|---|
| BLEU4 | 0,08 | 0,08 | 0,06 | 0,06 | 0,07 |
| BLEU3 | 0,19 | 0,187 | 0,155 | 0,151 | 0,166 |
| BLEU2 | 0,287 | 0,276 | 0,221 | 0,235 | 0,257 |
| BLEU1 | 0,55 | 0,514 | 0,462 | 0,5 | 0,572 |

*Ilustración 33. Evolución de la métrica BLEU para el ensayo 5 en función del número de épocas.*




| | 1 | 25 | 50 | 75 | 100 |
|---|---|---|---|---|---|
| ROUGE-L | 0,35 | 0,352 | 0,28 | 0,4 | 0,41 |
| METEOR | 0,187 | 0,152 | 0,162 | 0,158 | 0,188 |

*Ilustración 34. Evolución de las métricas ROUGE-L y METEOR para el ensayo 5 en función del número de épocas.*

En este último ensayo se aprecia que se ha conseguido minimizar el efecto del sobreajuste, ya que para un elevado número de épocas el rendimiento del modelo obtenido sigue mejorando. Puesto que para las tres métricas se han obtenido valores máximos con este valor de dropout, se escoge como modelo definitivo para realizar las pruebas con la Raspberry Pi el modelo entrenado durante 100 épocas con un 50% de dropout.

## 5.3 Muestra de descripciones de ejemplo

En esta sección se muestran algunos ejemplos de las descripciones generadas para algunos fragmentos de vídeos determinados. Como se puede apreciar, en algunos casos el modelo no describe con exactitud lo que ocurre o identifica correctamente los objetos, aunque el rendimiento obtenido para las limitaciones que ha tenido el proyecto es bastante óptimo.

| Imagen | Descripción generada |
|---|---|
|  | black dog is running through the grass |
|  | two girls are playing on the water |
|  | two dogs are playing in the grass |
|  | two men are playing on the grass |

| | |
|---|---|
|  | young boy in red shirt is playing on the water |
|  | man is riding bike on the water |

*Tabla 15. Ejemplos de descripciones generadas por el modelo implementado.*

## 5.4 Implementación Raspberry Pi 4

Finalmente, con el modelo entrenado y seleccionado en base a los criterios definidos en el apartado 5.2 Análisis de resultados obtenidos, se ha implementado un programa de descripción de vídeo en la Raspberry Pi.

El algoritmo diseñado permite acceder al módulo de cámara instalado en la Raspberry Pi para captar el vídeo y generar las descripciones de lo que está ocurriendo. Adicionalmente, una vez se han creado las descripciones, estas se guardan en formato de fichero de voz gracias a una de las funciones de la librería *pyttsx3* que permite convertir el texto en discurso de voz y guardar el resultado en formato mp3.

A continuación, se muestra en la Ilustración 35 el montaje del dispositivo Raspberry Pi 4, con su módulo de cámara conectado, la salida HDMI, la fuente de alimentación y el ventilador.

*Ilustración 35. Montaje de la Raspberry Pi del proyecto.*

A la hora de iniciar la generación de descripciones de texto, aparece el mensaje que se encuentra en la Ilustración 36.

```
Allocation of 411041792 exceeds 10% of system memory.
Allocation of 411041792 exceeds 10% of system memory.
```

*Ilustración 36. Alerta de sobrecarga de memoria en Raspberry Pi.*

Este mensaje de advertencia aparece cada vez que se carga el modelo de redes neuronales convolucionales VGG16, aunque gracias a que la Raspberry Pi ha sido configurada para emplear la totalidad de su memoria, se permite ejecutar el proceso. Debido a que el procesamiento de la imagen en tiempo real es demasiado lento para esta aplicación por las limitaciones de memoria de la Raspberry Pi, se ha decidido captar video durante unos segundos hasta que se presione una tecla prefijada del teclado para indicar que se para la captación de video, almacenando dicho video en el directorio raíz. Una vez hecho lo anterior, se descompone el vídeo en fotogramas internamente y se procesan, generando una descripción para cada fotograma y almacenando el resultado en el fichero de voz correspondiente. Para esta implementación se ha escogido una velocidad de captación de vídeo de 1 fotograma cada 4 segundos.

A continuación, se muestra un ejemplo de varios fotogramas extraídos de la captación de una secuencia de vídeo y su descripción:

| Imagen | Descripción |
|---|---|
|  | two girls are sitting on the street |
|  | man in red shirt is sitting on the water |
|  | man in red shirt is sitting on the street |

*Tabla 16. Ejemplo captación de la secuencia de vídeo y su descripción generada.*

El archivo generado con la lectura de las descripciones se almacena en el mismo directorio donde se encuentra el programa, como se muestra en la Ilustración 37.

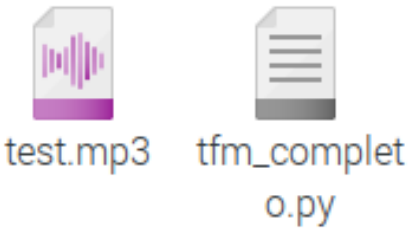



*Ilustración 37. Ejemplo de almacenamiento de un fichero de test con las descripciones generadas.*

Como puede apreciarse, las descripciones generadas no son tan precisas y la calidad de la cámara es relativamente baja.

## 5.5 Análisis temporal de los resultados

Una vez ajustado el tiempo de captura a la capacidad de la Raspberry Pi, se lleva a cabo un estudio de los tiempos necesarios para la totalidad del proceso. El resumen de los tiempos medidos necesarios para llevar a cabo una ejecución se muestra en la Tabla 17.

| Concepto | Tiempo requerido |
|---|---|
| **Tiempo de carga** | 14.3 s |
| **Captura de vídeo** | 12 s |
| **Generación primera descripción** | 21.7 s |
| **Generación segunda descripción** | 18 s |
| **Generación tercera descripción** | 19.3 s |
| **Generación fichero de voz** | 15 s |
| **Tiempo total** | 1 minuto 40 segundos |

*Tabla 17. Descomposición del tiempo empleado para la ejecución del programa de descripción de vídeo.*

Los resultados reflejados en la tabla anterior respaldan el hecho de que con el equipo actual no se puede llevar a cabo una descripción de vídeo en tiempo real en sentido estricto. El límite de fotogramas a procesar reduce la duración del vídeo captado a 12 segundos, necesitando 1 minuto y 40 segundos en total para completar toda la ejecución. Esto significa que el proceso emplea únicamente el 12% del tiempo requerido en captar el vídeo, el resto se emplea en procesar y emitir el resultado. Toda esto queda reflejado de manera visual en el gráfico mostrado en la Ilustración 38.

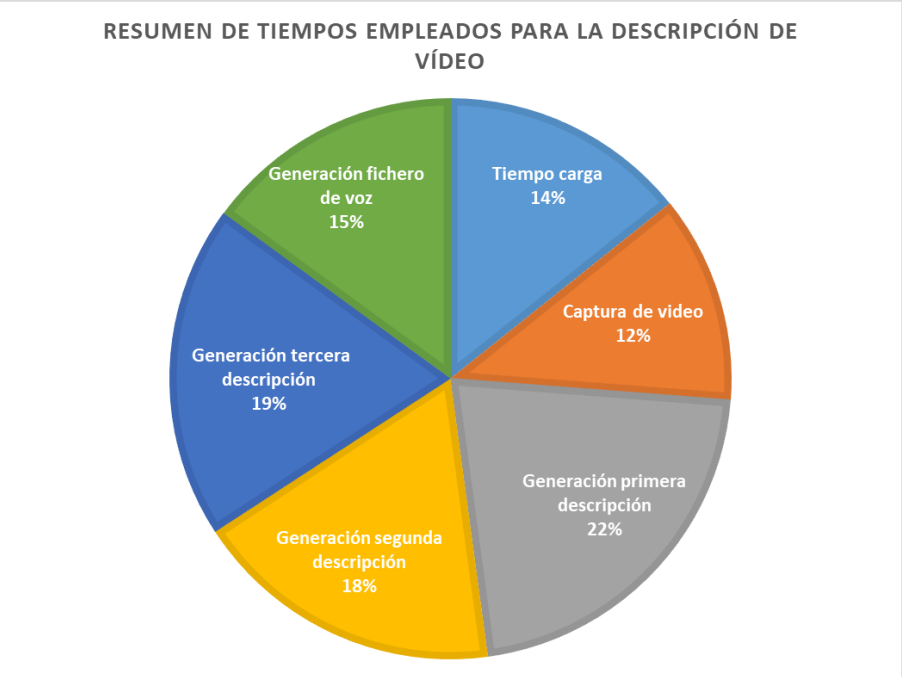



*Ilustración 38. Desglose del tiempo empleado para una ejecución del programa de descripción de vídeo.*

De esta manera se concluye que se necesita 7.3 veces más tiempo en procesar la información, que en captar el vídeo en sí mismo. Esto es debido a la limitación de capacidad de cómputo del dispositivo empleado, que introduce un cuello de botella e impide que la descripción de vídeo se pueda realizar en tiempo real. Puesto que la capacidad de cómputo supone un impedimento para que este dispositivo se use como solución para personas invidentes, se puede plantear la posibilidad de implantar este proyecto en un dispositivo más adecuado, aunque los límites de presupuesto y de duración por tratarse de un proyecto académico lo impidan en este caso.

# 6 Conclusiones y líneas futuras

En este proyecto se ha llevado a cabo una labor de investigación en el ámbito de la descripción de vídeo empleando modelos de redes neuronales artificiales. Una vez documentado el estado del arte (Ver apartado "1.3 Estado del arte") de esta tecnología, se concluye que se debe emplear para este proyecto un modelo compuesto por una red neuronal convolucional seguida de una red neuronal recurrente. La red neuronal convolucional es la encargada de extraer las características de la información contenida en los fotogramas obtenidos por la descomposición del vídeo de entrada. Por otra parte, esta información contenida en las características se hace llegar a la red neuronal recurrente, donde se codifica junto a la secuencia de texto. Por último, a la salida del modelo de red neuronal recurrente se lleva a cabo la decodificación de la información, dónde se obtiene la secuencia de texto que describe la información visual introducida.

Una vez justificada la elección de la arquitectura del modelo implementado, se procede a la selección de la base de datos con la que se entrena para adquirir el conocimiento necesario para las tareas de descripción de vídeo. Después de evaluar las distintas bases de datos presentes en la literatura, resumidas en el apartado "3.1 Estudio de Bases de Datos para *video captioning*", se escoge finalmente la base de datos MSR-VTT para el entrenamiento del modelo de este proyecto. A la hora de tratar los datos presentes en MSR-VTT se observa que no se encuentran en el formato deseado, por lo que ha sido necesaria una etapa previa de depuración para adaptarlos a la aplicación propuesta. Esto supone un problema a la hora de universalizar la solución propuesta, ya que cada base de datos requiere de una etapa previa de acondicionamiento al formato requerido.

A continuación, con la base de datos seleccionada y la arquitectura propuesta, se han analizado los requerimientos computaciones del proyecto y seleccionado el entorno de programación más adecuado. Para las primeras etapas del proyecto, se ha empleado el entorno de programación en la nube Google Colaboratory (Ver apartado "3.3 Google Colaboratory"). En este entorno se ha llevado a cabo el acondicionamiento de la base de datos para el proceso de entrenamiento. Debido a la limitación temporal de uso, el proceso de entrenamiento se ha ejecutado en un entorno de programación local en el ordenador del autor de este proyecto. Una vez lanzado este proceso, se apreció que resultaba demasiado largo, llegando la duración del proceso a un mes si se empleaba la base de datos completa. Puesto que el carácter de este proyecto es puramente académico, se decidió acortar el tiempo de entrenamiento empleando algunas estrategias como la reducción de los vídeos a analizar y la reducción del número de palabras presentes en el vocabulario del modelo. Adicionalmente, se instalaron las librerías necesarias para que el proceso se pudiera acelerar mediante el motor gráfico del equipo, lo que permitía acortar notablemente el tiempo necesario de entrenamiento. Sin la ayuda del motor gráfico, la CPU alcanzaba un 99% de uso, y una vez utilizado éste, se reducía el uso de la CPU a un 47% mientras que la GPU empleaba un 19% de su capacidad. El conjunto de estrategias mencionadas ha permitido disminuir el tiempo de entrenamiento en un 30%.

Después de analizar los requerimientos computaciones, se lanza el proceso de entrenamiento con la carga progresiva de datos para solventar la falta de la capacidad de memoria RAM suficiente. Se proponen distintas configuraciones de la arquitectura propuesta, realizando los ensayos que se detallan en el apartado "5.2 Análisis de resultados obtenidos". Para la evaluación de estos resultados se han tenido en cuenta las métricas BLEU, METEOR, y ROUGE-L. El modelo que ofrece el mejor rendimiento basado en las métricas descritas es el que ha sido entrenado durante 100 épocas con un dropout del 50%, obteniendo la puntuación que se muestra en la Tabla 18.

| Ensayo | Dropout | Épocas | BLEU1 | BLEU2 | BLEU3 | BLEU4 | ROUGE-L | METEOR |
|---|---|---|---|---|---|---|---|---|
| 5 | 50% | 100 | 0,572 | 0,257 | 0,166 | 0,07 | 0,41 | 0,188 |

*Tabla 18. Puntuación de las métricas del modelo escogido.*

La implementación del modelo escogido se lleva a cabo en un dispositivo Raspberry Pi 4. Para cargar el modelo entrenado ha sido necesario extender la capacidad de cómputo de este dispositivo, ya que por defecto viene limitada y no era suficiente para esta aplicación. Después de realizar un análisis técnico de las limitaciones del dispositivo a la hora de llevar a cabo la tarea de descripción de vídeo, se llega a la conclusión de que el límite de memoria de la Raspberry Pi restringe la capacidad de fotogramas que se pueden procesar e impone que solamente sea posible procesar 3 fotogramas por cada ejecución del programa diseñado. Por último, el análisis temporal desarrollado en el apartado "5.5 Análisis temporal de los resultados" justifica que la Raspberry Pi no es el dispositivo más adecuado para la descripción de vídeo en tiempo real.

Con todo lo expuesto anteriormente en esta sección, se concluye que se han alcanzado los objetivos definidos al inicio del proyecto, definidos en la sección "1.4 Objetivos del trabajo".

Respecto a las líneas futuras, se podría plantear la posibilidad de emplear un modelo distinto a VGG16 para la extracción de características de las imágenes, siempre que no se cuente con la limitación de la capacidad de cómputo. Existen algunos modelos más pesados que han demostrado tener un mejor rendimiento, como por ejemplo Inception. Aunque el vocabulario se ha reducido de tamaño para incrementar el rendimiento del modelo, sería interesante estudiar la posibilidad de eliminar palabras o descripciones que aparezcan en muy pocas ocasiones. De esta manera, sería posible optimizar el modelo ya que únicamente predeciría las palabras que ha podido aprender con cierta garantía.

Adicionalmente, puede plantearse la posibilidad de emplear vectores con palabras codificadas que ya hayan sido entrenados previamente. Esto permite que el vocabulario sea más extenso por lo que si se asocia a una buena base de datos, se puede obtener un modelo con mayor rendimiento y más riqueza léxica en sus predicciones.

Por último, se pueden plantear arquitecturas o modelos distintos para comparar con los resultados obtenidos en este proyecto. En cualquier caso, la descripción de vídeo y la inteligencia artificial son campos que están en constante desarrollo, cualquier innovación puede ser introducida y evaluada.

# 7 Bibliografía

[1] "Ceguera y discapacidad visual." [Online]. Available: https://www.who.int/es/news-room/fact-sheets/detail/blindness-and-visual-impairment. [Accessed: 19-Jun-2020].

[2] "Informe sobre la ceguera en España." SEEOF

[3] B. Z. Yao, X. Yang, L. Lin, W. Lee, and S.-C. Zhu, "I2T: Image Parsing to Text Description." Proceedings of the IEEE, 98(8):1485–1508, 2010.

[4] A. Farhadi *et al.*, "Every picture tells a story: Generating sentences from images," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2010, vol. 6314 LNCS, no. PART 4, pp. 15–29.

[5] S. Li, G. Kulkarni, T. L. Berg, A. C. Berg, and Y. Choi, "Composing Simple Image Descriptions using Web-scale N-grams," Association for Computational Linguistics, 2011.

[6] G. Kulkarni *et al.*, "BabyTalk: Understanding and Generating Simple Image Descriptions." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011.

[7] H. Yu and J. M. Siskind, "Grounded Language Learning from Video Described with Sentences," In Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), 2013.

[8] M. Rohrbach, W. Qiu, I. Titov, S. Thater, M. Pinkal, and B. Schiele, "Translating Video Content to Natural Language Descriptions." In IEEE International Conference on Computer Vision (ICCV), 2013.

[9] P. Das, C. Xu, R. F. Doell, and J. J. Corso, "A Thousand Frames in Just a Few Words: Lingual Description of Videos through Latent Topics and Sparse Object Stitching." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013.

[10] P. Das, R. K. Srihari, and J. J. Corso, "Translating related words to videos and back through latent topics," in *WSDM 2013 - Proceedings of the 6th ACM International Conference on Web Search and Data Mining*, 2013, pp. 485–494.

[11] F. Metze, D. Ding, E. Younessian, and A. Hauptmann, "Beyond audio and video retrieval: topic-oriented multimedia summarization," *International Journal of Multimedia Information Retrieval*, vol. 2, no. 2, pp. 131–144, Jun. 2013.

[12] J. Thomason, S. Venugopalan, S. Guadarrama, K. Saenko, and R. Mooney, "Integrating Language and Vision to Generate Natural Language Descriptions of Videos in the Wild." In International Conference on Computational Linguistics (COLING), 2014.

[13] S. Guadarrama *et al.*, "Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2712–2719.

[14] N. Krishnamoorthy, G. Malkarnenkar, R. Mooney, K. Saenko, and S. Guadarrama, "Generating Natural-Language Video Descriptions Using Text-Mined Knowledge." In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2013

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," In Advances in Neural Information Processing Systems (NIPS), 2012.

[16] I. Sutskever Google, O. Vinyals Google, and Q. V Le Google, “Sequence to Sequence Learning with Neural Networks.” In Advances in Neural Information Processing Systems (NIPS), pages 3104–3112, 2014.

[17] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko, “Translating Videos to Natural Language Using Deep Recurrent Neural Networks,” In Proceedings of Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT), 2015.

[18] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, “Sequence to Sequence-Video to Text.” In IEEE International Conference on Computer Vision (ICCV), 2015.

[19] S. Venugopalan, L. A. Hendricks, R. Mooney, and K. Saenko, “Improving LSTM-based Video Description with Linguistic Knowledge Mined from Text.” Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2016.

[20] N. Aafaq, A. Mian, W. Liu, S. Z. Gilani, and M. Shah, “Video description: A survey of methods, datasets, and evaluation metrics,” *ACM Computing Surveys*, vol. 52, no. 6, pp. 1–28, 2019.

[21] J. Donahue *et al.*, “Long-term recurrent convolutional networks for visual recognition and description,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, no. June, pp. 2625–2634, 2015.

[22] M. Rohrbach, S. Thater, W. Qiu, M. Pinkal, I. Titov, and M. Planck, “Translating Video Content to Natural Language Descriptions" *Iccv*, no. December, pp. 433–440, 2013.

[23] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” *Advances in Neural Information Processing Systems*, vol. 4, no. January, pp. 3104–3112, 2014.

[24] N. Srivastava, “Understanding online behaviour - ProQuest,” *International Conference on Machine Learning (ICML)*, vol. 37, 2015.

[25] H. Yu, J. Wang, Z. Huang, Y. Yang, and W. Xu, “Video paragraph captioning using hierarchical recurrent neural networks,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 4584–4593, 2016.

[26] V. Mnih *et al.*, “Playing Atari with Deep Reinforcement Learning,” pp. 1–9, 2013.

[27] V. Mnih *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[28] X. Wang, W. Chen, J. Wu, Y.-F. Wang, and W. Y. Wang, “Video Captioning via Hierarchical Reinforcement Learning.” arXiv:1711.11135, (2017).

[29] 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*, 2016, vol. 2016-December, pp. 770–778.

[30] Y. Chen, S. Wang, W. Zhang, and Q. Huang, “Less Is More: Picking Informative Frames for Video Captioning.” arXiv:1803.01457, (2018).

[31] “MyEye 2 Specification - OrCam.” [Online]. Available: https://www.orcam.com/es/myeye2/specification/. [Accessed: 26-Feb-2021].

[32] “Tech Specs – Glass.” [Online]. Available: https://www.google.com/glass/tech-specs/#tech-specs-masonry-dialog-0. [Accessed: 27-Feb-2021].

[33] X. Goldberg, "Introduction to semi-supervised learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6, pp. 1–116, Jun. 2009.

[34] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives." IEEE Transactions on Software Engineering 35(8):1798-1828, 2013.

[35] F. López-Muñoz, J. Boya, and C. Alamo, "Neuron theory, the cornerstone of neuroscience, on the centenary of the Nobel Prize award to Santiago Ramón y Cajal," *Brain Research Bulletin*, vol. 70, no. 4–6, pp. 391–405, Oct. 2006.

[36] "Aprendiendo acerca de Redes Neuronales Artificiales | by Jorge Martinez | Maule Devs | Medium." [Online]. Available: https://medium.com/maule-devs/aprendiendo-acerca-de-redes-neuronales-artificiales-5c81adbbe7ce. [Accessed: 18-Aug-2020].

[37] "Capítulos de Anatomía del Dr. Tulp: El entramado nervioso | Nervioso, Neuronas sinapsis, Neurotransmisores." [Online]. Available: https://www.pinterest.es/pin/406520303840586068/. [Accessed: 18-Aug-2020].

[38] *Artificial Neural Networks as Models of Neural Information Processing | Frontiers Research Topic*. Front. Comput. Neurosci. doi: 10.3389/fncom.2017.00114, 2017.

[39] A. Golfe, S. Martín, Y. Y. Lin, G. Prats, B. Javier, and M. Cabo, "Desarrollo e implementación de una aplicación para la caracterización, el procesado y análisis de señales electrohisterográficas en entorno MATLAB." 2018.

[40] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," 15(56):1929−1958, 2014.

[41] "CNN-RNA Convolucional – Numerentur.org." [Online]. Available: http://numerentur.org/convolucionales/. [Accessed: 28-Jun-2020].

[42] "Convolución - Wikipedia, la enciclopedia libre." [Online]. Available: https://es.wikipedia.org/wiki/Convolución. [Accessed: 28-Jun-2020].

[43] "Types of Convolution Kernels : Simplified - Towards Data Science." [Online]. Available: https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37. [Accessed: 28-Jun-2020].

[44] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," arXiv:1603.07285. Mar. 2016.

[45] "Redes Neuronales." [Online]. Available: https://ml4a.github.io/ml4a/es/neural_networks/. [Accessed: 28-Jun-2020].

[46] P. Liu, X. Qiu, and X. Huang, "Recurrent Neural Network for Text Classification with Multi-Task Learning." arXiv:1605.05101, IJCAI May 2016.

[47] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A Critical Review of Recurrent Neural Networks for Sequence Learning," arXiv:1506.00019 2015.

[48] G. Wong and G. Wong, *Build your own prototype Raspberry Pi minicomputer*. ubergizmo, 2011. [Online]. Avalaible: https://www.ubergizmo.com/2011/10/build-raspberry-pi-minicomputer/ [Accessed: 4-July-2020]

[49] "Raspberry Pi 4 Modelo B 4GB | PcComponentes.com." [Online]. Available: https://www.pccomponentes.com/raspberry-pi-4-modelo-b-4gb. [Accessed: 09-Aug-2020].

[50] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–14, 2015.

[51] “VGG16 - Convolutional Network for Classification and Detection.” [Online]. Available: https://neurohive.io/en/popular-networks/vgg16/. [Accessed: 13-Aug-2020].

# II. PRESUPUESTO

# 1. Desarrollo del proyecto

| Desarrollo del proyecto | | | | |
|---|---|---|---|---|
| **Descripción** | Unidad | Medición | Precio unitario | Importe |
| **Curso Deep learning aplicado al análisis de señales e imágenes** | h | 40 | 7.5€/h | 300€ |
| **Desarrollo del modelo de redes neuronales e implementación en Raspberry Pi 4** | h | 200 | 30h/h | 6000€ |
| **Documentación y elaboración de la memoria** | h | 60 | 15h/h | 900€ |
| **Costes directos** | | | | 7200€ |
| **Costes directos complementarios (2%)** | | | | 144€ |
| **Total** | | | | **7344€** |

*Tabla 19. Coste del desarrollo del proyecto.*

1. Curso Deep learning aplicado al análisis de señales e imágenes

Para llevar a cabo este proyecto ha sido necesario un periodo de formación para comprender tanto algunos conceptos de Deep learning como la implementación de modelos de redes neuronales. El curso necesario se ha realizado en el Centro de Formación Permanente de la Universidad Politécnica de Valencia, su nombre es “Deep learning aplicado al análisis de señales e imágenes” y fue impartido por Adrián Colomer Granero y Valery Naranjo Ornedo.

2. Desarrollo del modelo de redes neuronales e implementación en Raspberry Pi 4

Esta parte corresponde al trabajo realizado por parte del ingeniero. Por ello, se estima un precio por hora de trabajo de 30 euros. Esta actividad es la que mayor coste comporta ya que es de la que depende en mayor medida el proyecto.

3. Documentación y elaboración de la memoria

Puesto que se trata de un trabajo básico de redacción, se estima un precio por hora de 15 euros.

## 2. Materiales y recursos empleados

| Materiales y recursos empleados | | | | |
|---|---|---|---|---|
| **Descripción** | Unidad | Medición | Precio unitario | Importe |
| **Ordenador portátil Asus TUF Gaming FX505GD-BQ137** | H | 300 | 0.07€/h | 21€ |
| **Microsoft Office 2019** | H | 60 | 0.012€/h | 0.72€ |
| **Raspberry Pi 4 Pack LABSISTS** | Unidad | 1 | 91,63€ | 91,63€ |
| **Cámara Noir Raspberry LABSISTS** | Unidad | 1 | 29,99€ | 29,99€ |
| **Licencia Google Drive 100 GB** | Unidad | 6 | 1,99€ | 11,94€ |
| **Costes directos** | | | | 155,28€ |
| **Costes directos complementarios (2%)** | | | | 3,1€ |
| **Total** | | | | **158,38€** |

*Tabla 20. Materiales y recursos empleados*

1. Ordenador portátil Asus TUF Gaming FX505GD-BQ137

El precio de adquisición de este equipo es de 850€, con una vida útil estimada para esta tecnología de 6 años. Puesto que un año tiene 250 días laborables, la vida útil de este equipo es 1500 días laborables. Suponiendo la jornada laboral de 8 horas, el total de horas laborables durante la vida útil del ordenador será de 12.000 horas. El precio unitario se calcula mediante el cociente del precio de compra del ordenador y las horas laborables para las cuales servirá en su vida útil. Esto resulta 0.07€/h.

2. Microsoft Office 2019 para estudiantes

Puesto que el precio de la licencia es de 149€ y su amortización se limitará a la vida útil del equipo, que es de 12000 horas laborales, el precio de uso de este programa por hora resulta de 0.012€/h.

3.Raspberry Pi 4 Pack LABSISTS

Equipo necesario para implementación del modelo y adquisición de imágenes, El precio que se muestra en la Tabla 20 no incluye el IVA.

4. Cámara Noir Raspberry Pi LABSISTS

Módulo de cámara compatible con Raspberry Pi que se empleará para la adquisición de vídeo.

5. Licencia Google Drive 100GB

Licencia necesaria para poder almacenar en Google Drive la cantidad de datos deseada.

## 3. Presupuesto total

El coste total del proyecto es el que se muestra a continuación:

| PRESUPUESTO TOTAL | |
|---|---|
| **Descripción** | Importe |
| **Desarrollo del proyecto** | 7344€ |
| **Materiales y recursos empleados** | 158,38€ |
| **Presupuesto de ejecución material** | 7502,38€ |
| **Beneficio industrial (6%)** | 450,14€ |
| **Gastos generales** | 975,30€ |
| **Presupuesto de ejecución por contrata** | 8927,82€ |
| **IVA (21%)** | 1874,82€ |
| **Coste total del proyecto** | **10802,64€** |

*Tabla 21. Presupuesto total*

El presupuesto total del proyecto es de DIEZ MIL OCHOCIENTOS DOS EUROS CON SESENTA Y CUATRO CÉNTIMOS.