UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Generación Automatizada de Descripciones de Imágenes mediante Inteligencia Artificial

Trabajo Fin de Grado

Grado en Ciencia de Datos

AUTOR/A: Miravet Tenés, Joan

Tutor/a: Casacuberta Nolla, Francisco

CURSO ACADÉMICO: 2023/2024

Resumen

Este Trabajo de Fin de se enfoca en el estudio e implementación de modelos para la generación automática de descripciones de imágenes, un área de la IA que une la visión por computador y el procesamiento del lenguaje natural. En primer lugar, se lleva a cabo una revisión de los trabajos estado del arte en este campo. A continuación, se proponen y desarrollan dos arquitecturas para abordar la tarea. La primera es una basada en el modelo codificador-decodificador, utilizando redes neuronales convolucionales (CNN) combinadas con redes neuronales recurrentes (LSTM). Esta metodología aprovecha la capacidad de las CNN para extraer características visuales de las imágenes y la habilidad de las LSTM para generar secuencias de texto descriptivas. La segunda arquitectura emplea modelos basados en *Transformers*, específicamente *Vision Transformer* (ViT) para la extracción de características visuales y GPT-2 para la generación de texto. Se proporciona una explicación detallada de los componentes de ambas arquitecturas. Finalmente, se implementan y evalúan ambas arquitecturas, comparando sus resultados para analizar las mejoras y avances que cada metodología aporta en la generación automática de descripciones.

Palabras clave: Deep learning, Visión por Computador, Procesamiento del Lenguaje Natural, Redes Neuronales, *Transformers*, Descripción de imágenes, Aprendizaje automático.

Resum

Aquest Treball de Fi de Grau es centra en l'estudi i la implementació de models per a la generació automàtica de descripcions d'imatges, una àrea de la IA que uneix la visió per computador i el processament del llenguatge natural. En primer lloc, es realitza una revisió dels treballs estat de l'art en aquest camp. A continuació, es proposen i desenvolupen dues arquitectures per afrontar la tasca. La primera es basa en el model codificador-decodificador, utilitzant xarxes neuronals convolucionals (CNN) combinades amb xarxes neuronals recurrents (LSTM). Aquesta metodologia aprofita la capacitat de les CNN per extreure característiques visuals de les imatges i l'habilitat de les LSTM per generar seqüències de text descriptives. La segona arquitectura utilitza models basats en *Transformers*, específicament Vision *Transformer* (ViT) per a l'extracció de característiques visuals i GPT-2 per a la generació de text. Es proporciona una explicació detallada dels components de totes dues arquitectures. Finalment, s'implementen i s'avaluen ambdues arquitectures, comparant els seus resultats per analitzar les millores i els avanços que cada metodologia aporta en la generació automàtica de descripcions.

Paraules clau: Deep learning, Visió per Computador, Processament del Llenguatge Natural, Xarxes Neuronals, *Transformers*, Descripció d'imatges, Aprenentatge automàtic.

Abstract

This Final Degree Project focuses on the study and implementation of image captioning models, an area of AI that combines computer vision and natural language processing. First, a thorough review of the state-of-the-art works in this field is conducted. Next, two architectures will be proposed and developed. The first is based on an encoder-decoder model, using Convolutional Neural Networks (CNNs) combined with Long Short-Term Memory networks (LSTMs). This methodology leverages the ability of CNNs to extract visual features from images and the capability of LSTMs to generate descriptive text sequences. The second architecture employs *Transformer*-based models, specifically Vision *Transformer* (ViT) for visual feature extraction and GPT-2 for text generation. A detailed explanation of the components of both architectures is provided. Finally, both architectures are implemented and evaluated, comparing their results to analyse the improvements and advancements each methodology brings to image captioning.

Keywords: Deep learning, Computer Vision, Natural Language Processing, Neural Networks, *Transformers*, Image Captioning, Machine Learning.

Tabla de contenidos

Índice de figuras

Índice de tablas

1. Introducción

En la era digital actual, la proliferación de imágenes es exponencial. Cada día, millones de imágenes se suben a internet y se almacenan en bases de datos, lo que genera un desafío significativo en términos de organización, búsqueda y accesibilidad. La descripción precisa y significativa de imágenes se ha convertido en una necesidad crítica para mejorar la indexación y búsqueda de imágenes en línea, facilitando así su acceso y utilización eficiente en diversos contextos, desde bibliotecas digitales hasta redes sociales y plataformas de comercio electrónico.

El aumento de dispositivos con cámaras, como teléfonos móviles o cámaras digitales, ha impulsado este gran volumen de imágenes digitales. Esto presenta desafíos únicos para las plataformas que deben gestionar y organizar estas imágenes de manera efectiva. La capacidad de buscar y recuperar imágenes relevantes de amplias colecciones depende en gran medida de la existencia de descripciones precisas y detalladas que puedan asociarse con cada imagen.

Tradicionalmente, la descripción de imágenes ha sido un proceso manual, llevado a cabo por humanos que analizan el contenido visual y proporcionan etiquetas o descripciones textuales. Este proceso, aunque efectivo en términos de calidad y relevancia contextual, presenta varios inconvenientes. En primer lugar, es laborioso y consume mucho tiempo, lo que limita su escalabilidad frente al crecimiento exponencial de imágenes. En segundo lugar, es subjetivo, ya que diferentes personas pueden interpretar el mismo contenido de manera distinta, lo que introduce variabilidad en las descripciones, pudiendo afectar a la calidad de los datos y a la eficacia de los sistemas de recuperación de imágenes.

En respuesta a estos desafíos, el campo de la Inteligencia Artificial (IA) ha avanzado hacia el desarrollo de sistemas capaces de generar descripciones de imágenes de manera automatizada. Estos sistemas utilizan técnicas avanzadas de procesamiento de imágenes y aprendizaje profundo para analizar y comprender el contenido visual, y luego generar descripciones textuales coherentes y contextualmente relevantes. Este enfoque no solo promete mejorar la eficiencia y consistencia en la descripción de imágenes, sino que también abre nuevas posibilidades para la aplicación de tecnologías de IA en la gestión de contenido visual.

1.1 Motivación

Como estudiante de Ciencia de Datos, a lo largo de estos años de estudio he profundizado sobre los sistemas de aprendizaje automático y he sido testigo de cómo estos han ido evolucionando recientemente y ganando popularidad, sobre todo por la

llegada de modelos de lenguaje avanzado como es el caso de *ChatGPT*[1]. Estos modelos no solo han demostrado grandes capacidades en la generación de texto y comprensión del lenguaje natural, sino que también han abierto nuevas puertas en aplicaciones de Inteligencia artificial.

En la actualidad, la inteligencia artificial se encuentra en un punto de inflexión. La rápida evolución de los modelos de lenguaje y las redes neuronales ha permitido avances significativos en la capacidad de las máquinas para comprender y generar lenguaje humano. La popularidad y el éxito de modelos como *ChatGPT* son testimonio del potencial de estas tecnologías. Sin embargo, uno de los desafíos más complejos es la integración de estas capacidades lingüísticas con el procesamiento de imágenes, con el objetivo de generar descripciones automáticas.

La idea de que una computadora pueda "reconocer" una imagen y generar una descripción precisa y contextualmente correcta me parece fascinante. Esta capacidad no solo representa un gran logro técnico, sino que también abre innumerables posibilidades para aplicaciones prácticas. Mi interés en este campo se debe a la combinación de su complejidad técnica y su potencial impacto en diversas áreas, desde la mejora de la accesibilidad hasta la optimización de motores de búsqueda.

En resumen, la situación actual de la inteligencia artificial, con sus rápidos avances y su creciente popularidad, me motiva a profundizar en esta área para desarrollar sistemas que puedan interpretar y describir imágenes de manera automatizada y precisa.

1.2 Objetivos

El objetivo principal de este Trabajo de Fin de Grado es el de investigar y desarrollar un algoritmo inteligente capaz de generar descripciones automáticas precisas y contextualmente relevantes para imágenes mediante el uso de técnicas avanzadas de Inteligencia Artificial (IA). Los principales objetivos son:

1. **Investigación y Estado del Arte**

   - Realizar un estudio exhaustivo de las técnicas actuales utilizadas para la generación automática de descripciones de imágenes.

   - Revisar y analizar las metodologías y algoritmos más relevantes en el campo.

2. **Desarrollo de un Sistema Prototipo**

   - Diseñar y desarrollar un sistema prototipo basado en técnicas avanzadas de procesamiento de imágenes y aprendizaje profundo.

[1] https://chatgpt.com/

3. **Evaluación y Validación**

   - Evaluar la efectividad y la calidad de las descripciones generadas por el sistema prototipo en comparación con métodos existentes o con descripciones humanas.

1.3 Estructura de la memoria

La memoria sigue la siguiente estructura:

- **Capítulo 1. Introducción.**

  En la introducción se presenta el trabajo, se explica la motivación detrás del desarrollo del trabajo y los objetivos propuestos.

- **Capítulo 2. Estado del arte.**

  En este capítulo, se realiza una revisión exhaustiva del estado del arte en la generación automática de descripciones de imágenes. Se analiza la evolución de los modelos desde sus primeras versiones hasta los enfoques más avanzados y efectivos disponibles hoy en día. Se examinan los principales hitos y avances en tecnología que han permitido mejorar la precisión y coherencia de las descripciones generadas, proporcionando un contexto para las contribuciones del presente trabajo.

- **Capítulo 3. Análisis del problema.**

  Este capítulo aborda los desafíos y las complejidades asociadas con la generación automática de descripciones de imágenes. Se identifican las principales dificultades que enfrentan los modelos actuales, incluyendo limitaciones en la comprensión de contenido visual y la generación de texto coherente.

- **Capítulo 4. Arquitecturas y Métodos en la Generación Automática de Descripciones.**

  Este capítulo detalla las arquitecturas utilizadas en la generación automática de descripciones de imágenes. Se inicia con los modelos basados en CNN + RNN, explicando cómo las CNN extraen características visuales y las LSTM generan texto descriptivo. Luego, se introduce el concepto de mecanismos de atención, que mejoran la capacidad del modelo para enfocar diferentes partes de la imagen. Finalmente, se examinan los modelos avanzados basados en *Transformers*, que han revolucionado el campo al proporcionar un procesamiento más preciso y eficiente.

- **Capítulo 5. Recursos utilizados**

Este capítulo detalla el entorno y lenguaje de programación utilizados, así como el conjunto de datos empleado para la experimentación.

- **Capítulo 6. Marco experimental**

En este apartado se describe el marco experimental para la implementación y evaluación de las dos arquitecturas propuestas. Se divide en secciones que cubren las bibliotecas utilizadas, el proceso de codificación y decodificación en la arquitectura CNN + LSTM, el preprocesado de datos y la configuración del modelo en la arquitectura basada en *Transformers*, y la presentación y análisis de los resultados obtenidos.

- **Capítulo 7. Conclusiones**

En el capítulo de conclusiones, se realizará un análisis crítico de los resultados obtenidos y se comparan los resultados entre ambas arquitecturas. También se relacionan los conocimientos obtenidos durante las diferentes asignaturas del grado y se valoran las competencias transversales marcadas por la UPV.

- **Capítulo 8. Trabajos futuros**

En este último capítulo se sugiere una línea de investigación futura.

2. Estado del arte

Hoy en día, la inteligencia artificial (IA) se ha convertido en una palabra de moda, vista por algunos como una promesa para resolver problemas complejos y por otros como una amenaza con el potencial de impactar negativamente en sus vidas. Sin embargo, a pesar de su popularidad actual, la IA no es una novedad reciente.

Su origen formal se remonta a mediados del siglo XX, con hitos como el primer estudio de redes neuronales artificiales de McCulloch y Pitts en 1943 [1], y la publicación de Alan Turing en 1950 [2] en la que planteó la famosa pregunta: "¿Pueden las máquinas pensar?" y donde introdujo el Test de Turing como criterio para medir la inteligencia de las máquinas.

El desarrollo de la visión por computador, un subcampo esencial de la IA, comenzó a ganar atención con el desarrollo del perceptrón en 1957 por Frank Rosenblatt [3]. Este fue uno de los primeros modelos de red neuronal capaz de aprender a reconocer patrones simples en datos visuales. Aunque el perceptrón estaba limitado a problemas linealmente separables, sentó las bases para futuras investigaciones en redes neuronales más complejas. En paralelo a estos avances, el procesamiento del lenguaje natural (NLP), otro componente crucial de la IA, comenzó a desarrollarse. Los primeros enfoques de NLP en los años 50 y 60 se centraron en técnicas basadas en reglas para el análisis del lenguaje y sistemas de traducción automática, como el Modelo Georgetown-IBM [4] en 1954, que tradujo frases del ruso al inglés.

En 1959, Arthur Samuel amplió los horizontes de la inteligencia artificial con su desarrollo del concepto de *machine learning*. Samuel, pionero en el campo, acuñó el término "aprendizaje automático" y creó un programa de ajedrez que podía aprender y mejorar su rendimiento a través de la experiencia, marcando uno de los primeros ejemplos prácticos de esta tecnología emergente [5].

Durante los años 80 y 90 se realizaron avances significativos. En el campo de la visión por computador, se desarrollaron redes neuronales multicapa (MLP) [6] y redes neuronales convolucionales (CNN) [7], que mejoraron el reconocimiento de patrones complejos y el análisis de datos visuales. En el ámbito del procesamiento del lenguaje natural (NLP), se introdujeron modelos estadísticos y técnicas de aprendizaje automático, como los modelos de n-gramas, que optimizaron la capacidad de las máquinas para modelar el lenguaje [8].

En la última década, la IA ha alcanzado un nuevo nivel de madurez, impulsada por el acceso a grandes volúmenes de datos (*big data*) [9], avances en hardware y la computación en la nube, dando lugar a la “primavera perpetua” en la que nos encontramos actualmente. En particular, el desarrollo de algoritmos de aprendizaje profundo ha permitido alcanzar niveles de precisión sin precedentes en aplicaciones complejas, como el reconocimiento de imágenes [10], la traducción automática [11][12] y el procesamiento del lenguaje natural [13][14].

A medida que el *machine learning* ha evolucionado, en los últimos años ha emergido el aprendizaje profundo o *deep learning* [15][16]. Esta técnica avanzada entrena redes neuronales artificiales con grandes conjuntos de datos, permitiéndoles aprender y tomar decisiones inteligentes e independientes. Estas redes han ganado popularidad debido a su capacidad para mejorar la precisión y el rendimiento con el tiempo, sin necesidad de programación explícita para cada tarea específica. Son especialmente adecuadas para desafíos complejos como el análisis de imágenes, donde los algoritmos deben identificar patrones y características intrincadas en los datos visuales [10][17].

En este contexto, las redes neuronales convolucionales (CNNs), capaces de aprender automáticamente características visuales complejas a partir de grandes volúmenes de datos, revolucionaron el campo [10]. Desde entonces, la visión por computador ha avanzado rápidamente, con aplicaciones que van desde la conducción autónoma hasta el diagnóstico médico, destacándose por su capacidad para procesar y entender datos visuales con una precisión sin precedentes. Estos avances han consolidado la visión por computador como una disciplina central dentro de la IA, permitiendo que la IA en su conjunto alcance nuevos niveles de madurez y aplicabilidad en el mundo real.

A principios del siglo XXI, el campo del procesamiento del lenguaje natural (NLP) también experimentó una transformación significativa con la llegada de técnicas basadas en redes neuronales y modelos de *embeddings* de palabras, como *Word2Vec* [18] y *GloVe* [19] Posteriormente, los modelos de *Transformers* como *BERT* [13] y *GPT* [14] llevaron el NLP a nuevas alturas, permitiendo avances importantes en generación de texto, comprensión de preguntas y respuestas, y traducción automática. De manera similar, los modelos de *Transformers* también han comenzado a revolucionar el campo de la visión por computador. Modelos como *Vision Transformer* (ViT) [20] han demostrado un rendimiento competitivo con las redes neuronales convolucionales (CNNs) en tareas complejas de análisis de imágenes, demostrando la versatilidad y la capacidad para aprender representaciones visuales a partir de grandes conjuntos de datos.

La combinación de visión por computador y procesamiento del lenguaje natural ha dado lugar a aplicaciones innovadoras, como la generación automática de descripciones de imágenes [21]. Esta técnica utiliza avanzadas metodologías de ambos campos para crear descripciones textuales precisas y coherentes basadas en el contenido visual de las imágenes. Modelos como *Show and Tell* [22], *Show, Attend and Tell* [23], *End-to-end Transformer based model for image captioning* [24], *Image Transformer* [25], y *CLIP* [26], entre otros, han sido fundamentales en este avance.

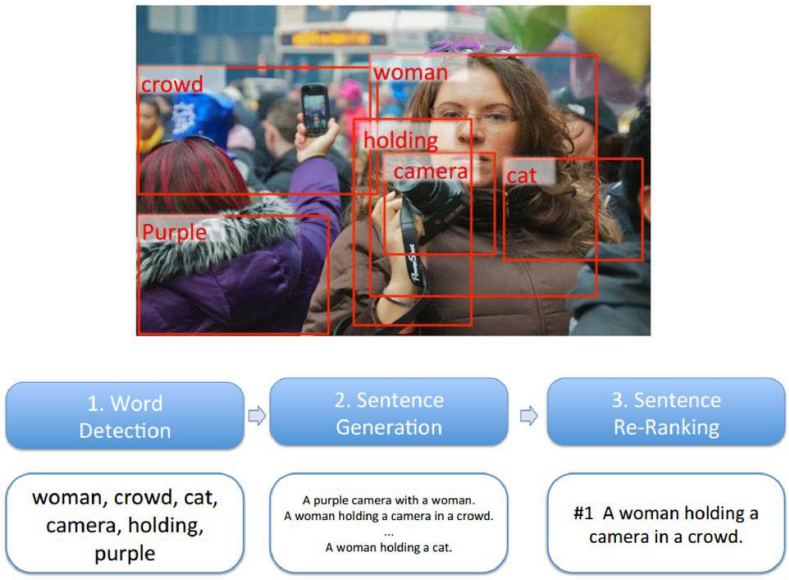

Ilustración 1: Modelo generador de descripciones. [27]

2.1 Generación de descripciones de imágenes

La generación automática de descripciones de imágenes es una aplicación avanzada del aprendizaje profundo que combina la visión por computador con el procesamiento del lenguaje natural (NLP). El proceso comienza con la extracción de características visuales de una imagen, lo que permite al sistema identificar y analizar detalles complejos como formas, texturas y colores. Una vez que se extraen estas características visuales, el siguiente paso es traducir la información en descripciones textuales coherentes y precisas. Este paso se logra mediante el uso de modelos de generación de lenguaje natural que son capaces de manejar secuencias de datos, generando así un texto que describe el contenido de la imagen de manera comprensible.

El estado actual del arte en esta área ha evolucionado significativamente gracias a los avances en aprendizaje profundo, especialmente con el desarrollo de arquitecturas de redes neuronales más sofisticadas y el empleo de grandes volúmenes de datos. Los enfoques principales utilizados para abordar esta tarea, son los siguientes:

- **Métodos Basados en Plantillas:** Este enfoque genera descripciones utilizando plantillas lingüísticas predefinidas.

- **Métodos Basados en Búsqueda:** Se fundamenta en localizar un subtítulo apropiado a partir de un conjunto de descripciones que corresponden a imágenes semánticamente similares.

- **Métodos Basados en Lenguaje:** Este enfoque se centra en utilizar modelos de lenguaje avanzados para generar descripciones de imágenes. Los métodos basados en lenguaje aprovechan técnicas de procesamiento del lenguaje natural para interpretar características visuales y convertirlas en descripciones textuales.

Estos métodos a menudo involucran redes neuronales profundas que combinan modelos de visión por computadora con modelos lingüísticos, permitiendo una descripción fluida y coherente del contenido visual. [21].

Estos últimos métodos, han demostrado obtener los mejores resultados. La función objetivo que guía el entrenamiento del modelo se define como sigue, donde θ representa los parámetros del modelo, I es una imagen dada, y S es la descripción ideal correspondiente:

$$\theta^* = argmax_\theta \sum_{(I,S)} log\ p(S|I;\theta)$$

El objetivo es encontrar los parámetros θ que maximizan la probabilidad logarítmica de la descripción ideal S dada la imagen I. A continuación, describo algunos de los enfoques y tendencias más destacados en el campo:

1. **Modelos Codificador-Decodificador basados en CNN-RNN:**

Inicialmente, los enfoques de la generación de descripciones de imágenes se basan en arquitecturas que combinaban redes neuronales convolucionales (CNN) para la extracción de características de imágenes con redes neuronales recurrentes (RNN), específicamente LSTMs (Long Short-Term Memory), para la generación de texto. Estos modelos son ejemplos clásicos de la arquitectura Codificador-Decodificador, donde la CNN actúa como el codificador que transforma la imagen en un vector de características, y la RNN funciona como el decodificador que traduce ese vector en una secuencia de palabras. Modelos como *Show and Tell* [22] son ejemplos de este enfoque.

2. **Modelos Atencionales:**

Los modelos con atención, como *Show, Attend and Tell* [23], mejoran las arquitecturas CNN-RNN al permitir que el modelo se enfoque en diferentes partes de la imagen mientras genera cada palabra de la descripción. En estos modelos Codificador-Decodificador con atención, el mecanismo de atención ayuda al decodificador a enfocarse en partes específicas de la entrada codificada, mejorando así la precisión y el contexto de las descripciones generadas. Estos modelos pueden producir descripciones más detalladas y contextualmente precisas, ya que pueden asignar más importancia a las áreas de la imagen que son relevantes para la palabra que se está generando en cada paso.

3. **Modelos Multimodales basados en *Transformers*:**

Con el auge de los *Transformers* como BERT [13] y GPT [14], el campo de la generación de descripciones de imágenes ha comenzado a adoptar estas arquitecturas. Modelos como *Image Caption Generation using Vision Transformer and GPT Architecture* [28] y ViLBERT [29] integran características visuales y textuales utilizando *Transformers*. Estos modelos se preentrenan en grandes conjuntos de datos de imagen-texto para aprender representaciones ricas, que luego se ajustan para tareas específicas de generación de descripciones. Los modelos CLIP (*Contrastive Language–Image Pretraining*) de OpenAI [26], *Flamingo* de *DeepMind* [30] y *Otter* [31], que amplía

el modelo multimodal LLaMA (*Large Language Model Meta AI*) de *Meta*[2] [32], también fusionan información visual y textual en un solo modelo, demostrando gran efectividad en tareas multimodales al alinear imágenes y textos durante el preentrenamiento.

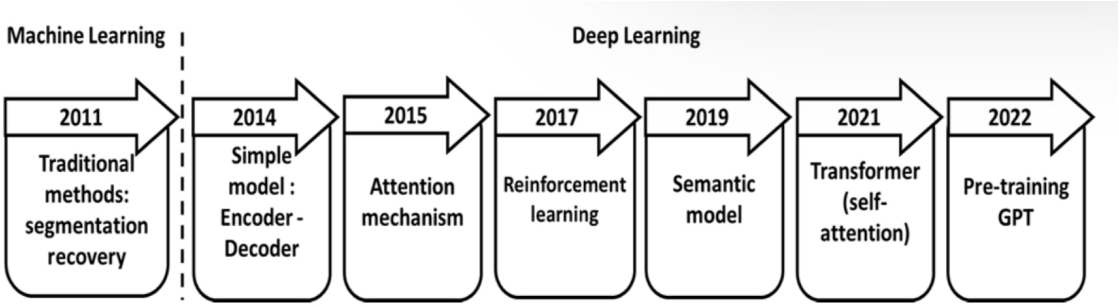

Ilustración 2: Evolución modelos generadores de descripciones. [33]

2.2 *Datasets*

Además de la evolución de los modelos, los avances en inteligencia artificial han sido impulsados por grandes conjuntos de datos etiquetados. Conjuntos de datos compuestos por imágenes junto a sus descripciones, en lenguaje natural.

Conjunto de datos	Imágenes	Textos	Artículo de Referencia
MS COCO	328K	5	Lin et al., 2014 [34]
Flickr30K	31K	5	Young et al., 2014 [35]
Flickr8K	8K	5	Hodosh et al., 2013 [36]
Pascal1K	1K	5	Rashtchian et al., 2010 [37]
Conceptual Captions	3,3M	1	Sharma et al., 2018 [38]
SBU Captioned Photo *Dataset*	1M	1	Ordondez et al., 2011 [39]
Visual Genome	108K	~50	Khrishna et al., 2017 [40]
IAPR TC-12	20K	1	Grubinger et al., 2006 [41]
nocaps	15,1K	~11	Agrawal et al., 2019 [42]

Tabla 1: Conjuntos de datos más conocidos

En el ámbito de la generación automática de descripciones de imágenes, tres conjuntos de datos destacan por encima de los demás: MS COCO, Flickr30K y Flickr8K.

[2] https://www.meta.com/es/

MS COCO (*Microsoft Common Objects in Context*): [34] Desarrollado por Microsoft en 2014, MS COCO es el conjunto de datos más utilizado para tareas de generación de descripciones. Contiene más de 328 mil imágenes que capturan escenas cotidianas complejas, cada una con anotaciones detalladas que incluyen descripciones en lenguaje natural, segmentaciones de objetos, y más. Cada imagen está acompañada por cinco descripciones textuales, proporcionando una amplia gama de perspectivas sobre la escena visual. En total, el conjunto de datos cuenta con más de 1.5 millones de anotaciones. En el primer lanzamiento del conjunto de datos incluía un total de 164 mil imágenes, divididas en conjuntos de entrenamiento, validación y prueba. De estas imágenes, 83000 se destinaron al conjunto de entrenamiento, 41000 al de validación y 4000 al de prueba. En 2015, se añadió un nuevo conjunto de prueba con 81000 imágenes, que incluía todas las imágenes del conjunto de prueba original junto con 40000 imágenes adicionales.

Flickr8K: [36] Este conjunto de datos está compuesto por 8091 imágenes obtenidas de la plataforma web de imágenes Flickr[3]. Las imágenes en Flickr8K abarcan una amplia variedad de escenas y actividades cotidianas. Cada imagen está acompañada por cinco descripciones textuales en inglés generadas por humanos, que ofrecen una variedad de detalles descriptivos. El conjunto se divide en 6000 imágenes para entrenamiento, 1000 para validación y 1000 para prueba.

Flickr30K: [35] La versión extendida de Flickr8K, incluye un total de 31793 imágenes. Este conjunto de datos proporciona una mayor diversidad y un número más amplio de ejemplos en comparación con Flickr8K. En Flickr30K, las imágenes se dividen en 29000 para entrenamiento, 1000 para validación y 1000 para prueba.

Otros conjuntos de datos, como *Conceptual Captions*, contienen un mayor número de imágenes, pero sus descripciones han sido generadas automáticamente en lugar de ser elaboradas por humanos. Por otro lado, *IAPR TC-12* se destaca por ofrecer descripciones en tres idiomas diferentes: alemán, inglés y español, aunque cada imagen está acompañada únicamente por una descripción en cada idioma.

2.3 Métricas de evaluación

En la evaluación de los modelos, es fundamental contar con métricas que puedan cuantificar de manera efectiva la calidad y precisión de las descripciones generadas. Dado que el objetivo principal de estos modelos es producir descripciones en lenguaje natural que sean coherentes y relevantes respecto al contenido visual, las métricas de evaluación deben capturar diversos aspectos de la calidad lingüística y la precisión descriptiva. Estas métricas pueden evaluar la exactitud semántica, la fluidez del texto generado, y la capacidad del modelo para capturar la esencia de las imágenes presentadas.

[3] https://www.flickr.com/

Entre las métricas más comúnmente utilizadas se encuentran BLEU (*Bilingual Evaluation Understudy*) [43], METEOR (*Metric for Evaluation of Translation with Explicit ORdering*) [44], ROUGE (*Recall-Oriented Understudy for Gisting Evaluation*) [45] y CIDEr (*Consensus-based Image Description Evaluation*) [46].

BLEU (Bilingual Evaluation Understudy): [43]

Es la métrica de evaluación más ampliamente utilizada para evaluar la calidad de las descripciones generadas por modelos de procesamiento de lenguaje natural. Aunque fue inicialmente diseñada para la evaluación de traducción automática, su aplicabilidad se ha extendido a diversas tareas, incluida la generación de descripciones de imágenes. BLEU ofrece una medida cuantitativa para valorar la calidad de las descripciones generadas al compararlas con un conjunto de descripciones de referencia.

Se basa en la comparación de n-gramas, que son secuencias de n palabras consecutivas dentro de un texto. Un n-grama puede considerarse una subsecuencia de una secuencia dada, y permite evaluar el grado de coincidencia entre las descripciones generadas y las de referencia. Por ejemplo, para la frase “Un niño juega a fútbol”, los unigramas (1-gramas) serían “Un”, “niño”, “juega”, “a”, “fútbol”, y sus bigramas (2-gramas) serían serían “Un niño”, “niño juega”, “juega al”, y “al fútbol”.

La métrica BLEU utiliza los unigramas, bigramas, trigramas (3-gramas) y tetragramas (4-gramas) de las frases, para calcular un puntaje que refleje cuán bien se alinea la descripción generada con las de referencia. Este enfoque permite una evaluación más precisa de la calidad y coherencia de las descripciones generadas. La puntuación BLEU se calcula en varios pasos. Primero, se calcula la precisión de n-gramas para diferentes valores de n. La precisión de n-gramas se define como la proporción de n-gramas que aparecen en la descripción generada en relación con el total de n-gramas presentes en la descripción de referencia.:

$$P_n = \frac{c(\delta(ng, nr))}{c(ng)}$$

Donde n puede ser 1, 2, 3, o 4, para un análisis en términos de unigramas, bigramas, trigramas y tetragramas, respectivamente. En este contexto,ng y nr se refiere a los n-gramas de la descripción generada y referencia respectivamente. La función delta $\delta(X, Y)$ devuelve 1 cuando los n-gramas son idénticos y 0 cuando son diferentes. Por último, $c(X)$ representa el número total de ocurrencias, lo que implica contar cuántos n-gramas son comunes entre la descripción generada y la de referencia en el numerador, y el número total de n-gramas en la descripción generada en el denominador.

Una vez obtenida la precisión, la puntuación BLEU para un valor de n se obtiene calculando la precisión de n-gramas para cada valor de n y luego combinándolos en una puntuación general utilizando una media ponderada geométrica:

$$BLEU = BP \cdot exp(\sum_{n=1}^{N} w_n log(P_n))$$

BP es la penalización por longitud (*brevity penalty*) y wn los pesos asignados a cada nivel de n-grama. Estos pesos suman a 1 y definen la importancia de cada tipo de n-grama en el cálculo final. Para BLEU-1, por ejemplo, el peso sería (1.00, 0.00, 0.00, 0.00), indicando que solo se considera la precisión de unigramas, $wn = \frac{1}{n}$.

$$BP = \begin{cases} 1 & si\ c > r \\ e^{(1-\frac{r}{c})} & si\ c \leq r \end{cases}$$

Donde c es la longitud de la descripción generada y r es la longitud de la descripción de referencia más cercana. La penalización ajusta la puntuación final para evitar que el modelo obtenga altas puntuaciones simplemente generando descripciones más cortas que las de referencia.

La puntuación BLEU final combina estas precisiones ponderadas y la penalización por longitud para proporcionar una medida integral de la calidad de las descripciones generadas. De esta manera, ofrece una evaluación precisa al comparar las descripciones generadas con las expectativas de referencia, proporcionando una métrica valiosa para evaluar la capacidad del modelo para producir descripciones precisas y coherentes.

METEOR (Metric for Evaluation of Translation with Explicit ORdering): [44]

METEOR fue desarrollada para superar algunas limitaciones de BLEU, especialmente en términos de sensibilidad a la variación de palabras y sinónimos. Esta métrica se basa en la coincidencia de palabras, frases y sinónimos, así como en la correspondencia de palabras en diferentes posiciones. También considera la estructura gramatical, buscando coincidencias entre las descripciones generadas y las referencias mediante técnicas como el *stemming* y el análisis de sinónimos. Esto permite que METEOR proporcione una evaluación más completa y flexible de la calidad del texto generado.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation): [45]

ROUGE es una familia de métricas que se centran en la capacidad de un modelo para generar contenido que sea relevante y exhaustivo en comparación con las descripciones de referencia. Se basan en el análisis de n-gramas, bigramas y trigramas para medir la cobertura y la recuperación del contenido entre las descripciones generadas y las de referencia. A diferencia de BLEU, que pone más énfasis en la precisión, ROUGE prioriza la capacidad del modelo para recuperar una cantidad significativa de contenido relevante, lo que lo hace especialmente útil para tareas de resumen y generación de texto más amplias.

CIDEr (Consensus-based Image Description Evaluation): [46]

CIDEr está diseñado específicamente para evaluar descripciones de imágenes. Este método mide la calidad de las descripciones generadas al comparar su consenso con un conjunto de descripciones de referencia. Utiliza un enfoque basado en la coincidencia de n-gramas, ponderando la importancia de las palabras y frases en función de su

relevancia en el conjunto de descripciones de referencia. CIDEr es particularmente útil para capturar la variabilidad en la forma en que se pueden describir las imágenes y para evaluar cómo bien las descripciones generadas coinciden con las expectativas humanas en un contexto más amplio.

2.4 Propuesta personal

Para la realización de este trabajo, se ha optado por utilizar el conjunto de datos *Flickr8K* y la métrica de evaluación BLEU para medir la calidad de las descripciones generadas. La selección de *Flickr8K* se debe a su tamaño manejable y a la alta calidad de las descripciones que ofrece, lo cual es ideal para entrenar y evaluar modelos de generación de texto sin incurrir en un costo computacional excesivo.

Por otro lado, BLEU ha sido elegida como la métrica de evaluación por su amplia aceptación en la comunidad de procesamiento de lenguaje natural y por su capacidad para ofrecer una medida cuantitativa de la calidad de las descripciones generadas. Aunque presenta algunas limitaciones, como su sensibilidad a la longitud de las frases y su falta de consideración de sinónimos y variaciones gramaticales, sigue siendo una herramienta valiosa para evaluar la calidad de las descripciones en tareas de generación automática de texto. Su enfoque en la precisión de los n-gramas proporciona una evaluación objetiva y comparativa que es fundamental para medir el rendimiento de los modelos en este ámbito.

Las variantes BLEU-1, BLEU-2, BLEU-3 y BLEU-4 se enfocan en evaluar la coincidencia de n-gramas entre un texto generado y uno de referencia, con cada variante proporcionando distintos niveles de precisión según la longitud de las secuencias de palabras.

- **BLEU-1** mide la coincidencia de unigramas (palabras individuales), siendo la métrica más básica. Evalúa si las palabras utilizadas en el texto generado están presentes en la referencia, pero sin tener en cuenta el contexto o el orden de las palabras.

- **BLEU-2** analiza la coincidencia de bigramas (secuencias de dos palabras consecutivas), lo que añade información sobre cómo se relacionan las palabras entre sí, aportando una evaluación más precisa que BLEU-1.

- **BLEU-3** va un paso más allá al considerar trigramas (secuencias de tres palabras), lo que permite valorar la coherencia en frases más largas, empezando a captar el flujo de la oración.

- **BLEU-4** ofrece la evaluación más detallada al analizar tetragramas (secuencias de cuatro palabras consecutivas), lo que permite capturar de manera más completa la estructura y el contexto de las oraciones.

Modelo	BLEU-1	BLEU-2	BLEU-3	BLEU-4
m-RNN (Mao et al., 2014) [47]	58.0	28.0	23.0	—
m-RNN (Mao et al., 2015b) [48]	56.5	38.6	25.6	17.0
m-RNN (Karpathy & Fei-Fei, 2015) [49]	57.9	38.3	24.5	16.0
Google NIC (Vinyals et al., 2014) [22]	63.0	41.0	27.0	—
Log Bilinear (Kiros et al., 2014) [50]	65.6	42.4	27.7	17.7
Emb-gLSTM (Jia et al., 2015) [51]	64.7	45.9	31.8	21.6
Soft-Attention (Xu et al., 2015) [23]	67.0	44.8	29.9	19.5
Hard-Attention (Xu et al., 2015) [23]	67.0	45.7	31.4	21.3
Wu et al. 2018 (Wu et al., 2017) [52]	74.0	54.0	38.0	27.0
ETransCap (Mundu et al., 2024) [53]	76.1	61.3	45.9	40.0

Tabla 2: Resultados obtenidos en trabajos estado del arte con Flickr8K

3. Análisis del problema

Para entender cómo un ordenador procesa una imagen, es importante reconocer que, a diferencia de los humanos, que perciben imágenes de forma global e intuitiva, las máquinas ven las imágenes como una colección de datos numéricos, específicamente como una matriz de píxeles. Cada píxel en una imagen digital tiene un valor numérico que representa su color, y la combinación de todos estos píxeles forma la imagen completa. En las imágenes en escala de grises, por ejemplo, cada píxel tiene un valor entre 0 (negro) y 255 (blanco). En las imágenes en color, cada píxel suele estar representado por tres valores (correspondientes a los colores rojo, verde y azul, o RGB), que juntos definen el color de ese píxel.

El desafío para un ordenador radica en interpretar estas matrices de números y extraer de ellas información semántica, como lo haría un ser humano al observar una imagen. Por ejemplo, para un humano, una imagen de un perro es instantáneamente reconocible como un perro. Para un ordenador, sin embargo, esa misma imagen es solo una matriz de números que necesita ser analizada y procesada para identificar patrones y características (como bordes, texturas, y formas) que permitan concluir que la imagen contiene un perro.

La dificultad radica en que la relación entre estos valores numéricos y el significado visual de la imagen no es directa ni evidente. Los algoritmos deben aprender a identificar y reconocer patrones complejos dentro de estos datos numéricos, lo que requiere técnicas avanzadas de procesamiento y modelado. Estas técnicas permiten a las máquinas descomponer la imagen en sus componentes básicos, como bordes y texturas, y luego combinarlos para formar una representación más abstracta y comprensible de la imagen, lo que constituye la base para tareas más complejas como la clasificación de imágenes o la generación de descripciones, que es el objetivo principal de este estudio.

Además del desafío de interpretar las matrices numéricas que componen una imagen, otro problema significativo es la generación de texto coherente a partir de esa información visual. No es suficiente que un modelo identifique objetos dentro de una imagen; también debe ser capaz de organizar esas observaciones en una secuencia lógica y comprensible de palabras que formen una descripción completa. Esto requiere que el modelo entienda el contexto y las relaciones entre los objetos, lo que implica generar texto que sea no solo gramaticalmente correcto, sino también coherente y relevante con respecto a los elementos visuales de la imagen.

3.1 Solución del problema

Para abordar este problema, primero se ha de buscar una forma de extraer las características significativas de una imagen Una vez que se obtienen estas características, el siguiente desafío es interpretar y darles sentido semántico. Para ello, se han desarrollado modelos Codificador-Decodificador, donde el componente codificador se encarga de extraer y representar las características visuales clave de la imagen, y el decodificador toma estas representaciones y las traduce en texto descriptivo. Este proceso permite generar descripciones coherentes y detalladas al combinar la información visual con el conocimiento lingüístico. En tiempos recientes, los modelos basados en *Transformers*, entrenados con grandes volúmenes de datos, han demostrado ser especialmente eficaces en esta tarea. Su capacidad para capturar relaciones complejas y contextos más amplios ha mejorado significativamente la calidad y precisión de las descripciones generadas.

La solución propuesta en el trabajo incluye la creación de dos modelos para la generación de descripciones automáticas de imágenes. El primer modelo es una arquitectura Codificador-Decodificador que combina redes neuronales convolucionales (CNN) con redes neuronales recurrentes LSTM. El segundo modelo utiliza *Transformers*, combinando Vision *Transformer* (ViT) para el procesamiento visual y GPT-2 para la generación de texto, logrando un procesamiento más eficiente y preciso. Ambos modelos se comparan para evaluar sus mejoras en la generación automática de descripciones. En las siguientes secciones se profundizará en sus diseños y funcionamiento.

4. Arquitecturas y Métodos en la Generación Automática de Descripciones

En esta sección se explicarán detalladamente cada uno de los componentes clave de los modelos que serán usados para la generación de descripciones automáticas de imágenes. Primero, se explorará el papel de las redes neuronales convolucionales (CNN), que se encargan de extraer características visuales esenciales de las imágenes. A continuación, se analizará el uso de redes neuronales recurrentes (RNN), específicamente LSTM, que procesan estas características para generar texto descriptivo. Luego, se discutirá el mecanismo de atención, que mejora la generación de texto permitiendo al modelo concentrarse en diferentes partes de la imagen durante el proceso de descripción. Finalmente, se abordará el uso de *Transformers*, una alternativa avanzada a las RNN, basada exclusivamente en mecanismos de atención.

4.1 Arquitecturas Codificador-Decodificador

Los modelos Codificador-Decodificador son una arquitectura fundamental en el aprendizaje automático, especialmente en tareas que requieren la conversión de una entrada en una salida secuencial, como la generación automática de descripciones de imágenes. Esta arquitectura se basa en dos componentes principales: el codificador y el decodificador, cada uno con funciones específicas en el procesamiento de datos.

El codificador se encarga de procesar la imagen y convertirla en una representación interna compacta y rica en información. El codificador suele estar compuesto por una Red Neuronal Convolucional (CNN) [7], que examina la imagen, extrayendo características visuales esenciales como formas, texturas y colores. Este proceso genera un vector de características que resume el contenido visual de la imagen de manera condensada. Recientemente, los Visual *Transformers* [20] han sustituido a las CNNs debido a su capacidad para capturar dependencias de largo alcance en los datos visuales a través de mecanismos de atención. El propósito del codificador es transformar la imagen en una representación que capture toda la información visual relevante, preparando así los datos para la siguiente fase del proceso.

El decodificador, por su parte, toma la representación generada por el codificador y la utiliza para producir una secuencia de palabras que conformen una descripción textual coherente de la imagen. Tradicionalmente, este componente ha utilizado Redes Neuronales Recurrentes (RNNs) o sus variantes, como LSTM (Long Short-Term Memory) [54], para generar el texto. Las RNNs procesan la información de manera secuencial, generando una palabra a la vez y manteniendo un estado interno que les permite recordar el contexto de palabras generadas previamente. Sin embargo, en desarrollos más recientes, los *Transformers* han reemplazado a las RNNs debido a su capacidad superior para manejar dependencias a largo plazo de manera más eficiente.

Los *Transformers* emplean mecanismos de atención que permiten enfocar diferentes partes del vector de características durante la generación del texto, mejorando la calidad y coherencia de las descripciones.

4.2 Codificador

4.2.1 Convolutional Neural Networks (CNN)

Las CNN han sido diseñadas específicamente para procesar datos con una estructura en forma de cuadrícula, como las imágenes, son capaces de aprender representaciones jerárquicas que capturan tanto características simples como complejas de los datos visuales. Están compuestas por varias capas (capas de entrada, convolucionales, de agrupamiento y completamente conectadas), cada una con un propósito específico para transformar y procesar los datos de entrada en representaciones abstractas más útiles para la clasificación o detección de objetos.

El proceso comienza con la capa de entrada, que recibe una imagen generalmente representada como una matriz tridimensional, donde las dimensiones corresponden al ancho, alto y canales de color de la imagen. Luego encontramos las capas convolucionales, el núcleo de las CNN. Estas aplican pequeños filtros o *kernels* que se desplazan sobre la imagen de entrada, realizando una operación de convolución que detecta características como bordes y texturas. Cada filtro se entrena para identificar un tipo específico de patrón o característica en la imagen. La operación de convolución produce un mapa de características (*feature map*). Este es una representación de la imagen original que resalta las características más importantes detectadas por los filtros aplicados durante la convolución. [79]

Por ejemplo, para una imagen en escala de grises, un filtro para detectar bordes horizontales podría ser:

-1	-1	-1
0	0	0
1	1	1

Cuando aplicamos este filtro a la imagen, realiza una operación de convolución que resalta las diferencias de intensidad entre filas, permitiendo identificar bordes horizontales. Si el filtro detecta un cambio brusco en la intensidad entre filas adyacentes (de -1 a 1), el resultado será un valor alto en el mapa de características, indicando la presencia de un borde horizontal en esa región. Una imagen con muchos bordes horizontales generará un mapa de características con valores altos en esas áreas.

Por otro lado, para detectar bordes verticales, la red podría utilizar un filtro como este:

-1	0	1
-1	0	1
-1	0	1

Matemáticamente, la operación de convolución entre un filtro *K* y la imagen de entrada *I* se define como:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n) \cdot K(i-m, j-n)$$

donde el símbolo " * " denota la operación de convolución. Esta operación consiste en aplicar el filtro K sobre la imagen de entrada I de manera que, en cada posición (i,j), se calcula una suma ponderada de los valores de los píxeles de I que se solapan con el filtro K. Esta suma ponderada da lugar a un nuevo valor $S(i,j)$ que se almacena en una matriz de salida.

- $I(m,n)$ representa los píxeles de la imagen de entrada en las posiciones (m,n).
- $K(i-m, j-n)$ son los pesos del filtro K que se aplican sobre los píxeles de I.
- $S(i,j)$ es el valor resultante en el mapa de características en la posición (i,j).

La convolución desplaza el filtro K por toda la imagen, multiplicando los valores de los píxeles de I por los correspondientes pesos del filtro y sumando los resultados. Esto permite detectar características específicas de la imagen en función de los valores del filtro.

Después de cada convolución, se aplica una función de activación para introducir no linealidad en el modelo. La función *ReLu* (*Rectified Linear Unit*) es la más común, esta convierte los valores negativos en cero y mantiene los valores positivos, lo que ayuda a que la red aprenda funciones complejas y mejora la capacidad del modelo para capturar patrones relevantes.

$$f(x) = max(0, x)$$

Cada uno de los mapas de características producidos como salida por la capa de convolución corresponde a una detección de patrones diferente realizada por diferentes filtros convolucionales. Estos sirven como entrada para las capas de agrupamiento o capas de *pooling* que se utilizan para simplificar la representación de los datos, manteniendo las características más importantes mientras se reducen las dimensiones del mapa de características. Esta reducción es crucial para controlar el tamaño de los datos a medida que se avanza en la red, evitando el crecimiento exponencial de las dimensiones. Al resumir la información más significativa, las capas de agrupamiento

garantizan que los patrones críticos detectados por las capas convolucionales se mantengan, ayudando también a prevenir el sobreajuste al reducir la complejidad del modelo.

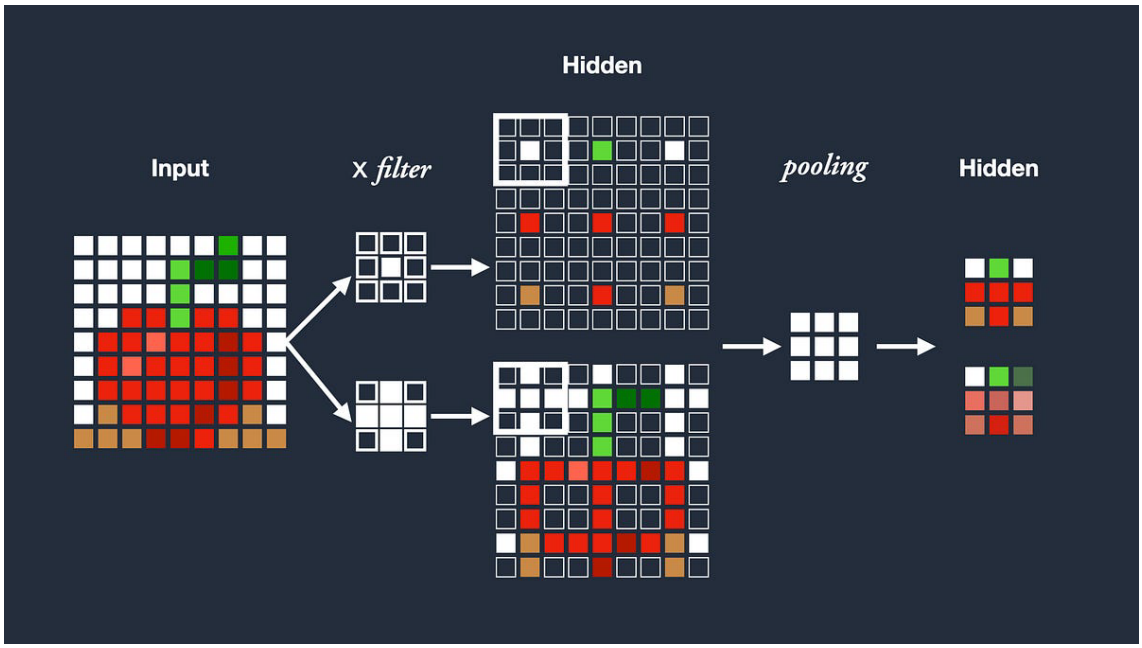

Ilustración 3: Capas convolucionales y de agrupamiento CNN. [55]

Hay diversas técnicas de agrupamiento, siendo las más comunes el *max pooling* y el *average pooling*. *Max pooling* toma el valor máximo de una región específica de los datos, destacando características prominentes como pueden ser bordes o picos de activación. *Average pooling* toma el valor promedio, proporcionando una representación más suavizada de los datos.

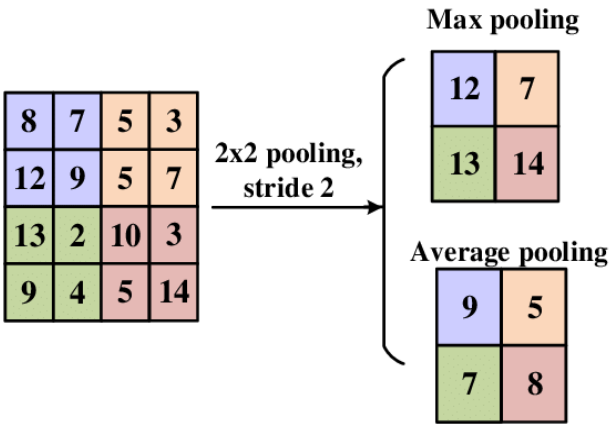

Ilustración 4: Técnicas de pooling. [56]

Una vez que las características visuales han sido extraídas y resumidas por las capas convolucionales y de agrupamiento, se aplanan y se pasan a través de una o más capas completamente conectadas. [56] En estas capas, cada nodo está conectado a todos los nodos de la capa anterior, permitiendo una combinación integral de la información visual

extraída. Este proceso transforma el vector de características visuales en una representación densa y rica, que captura los aspectos más relevantes de la imagen. Esta representación es luego utilizada para generar una salida que puede ser interpretada por modelos de lenguaje para producir texto descriptivo.

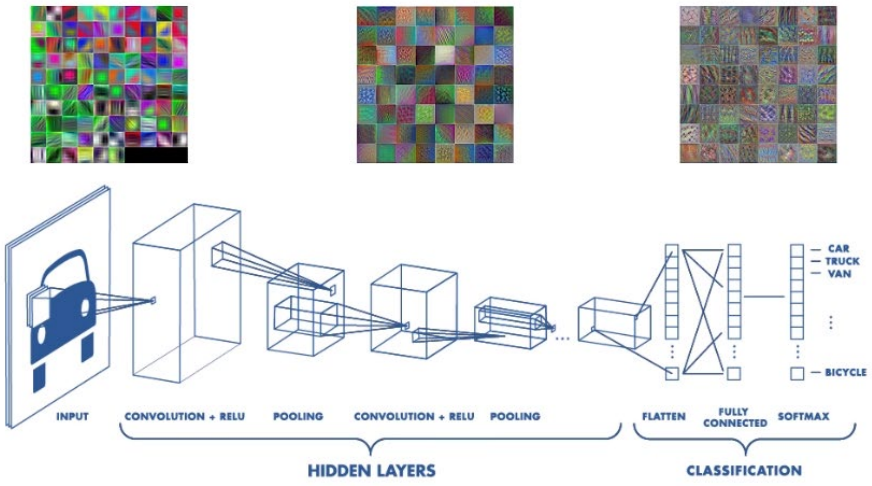

Ilustración 5: Esquema de funcionamiento de una CNN. [57]

Para utilizar una CNN en la extracción de características, se realiza una modificación en su estructura y se elimina la última capa del modelo, que normalmente es una capa de clasificación. Esta capa final es la responsable de tomar el vector de características generado por las capas anteriores y producir una predicción de la clase a la que pertenece la imagen (por ejemplo, si es una imagen de un perro o un gato).

Sin embargo, cuando el objetivo es utilizar la CNN únicamente como un extractor de características, lo que interesa es el vector que contiene la representación de alto nivel de la imagen, no la clasificación final. Por lo tanto, al remover la última capa de clasificación, se obtiene directamente este vector de características.

4.2.2 Modelos preentrenados

Entrenar una red neuronal convolucional (CNN) desde cero es un proceso complejo y costoso, ya que la red debe aprender a reconocer patrones visuales y ajustar una gran cantidad de parámetros para hacerlo de manera precisa. Este proceso requiere una gran cantidad de datos y un considerable poder computacional para ajustar todos los parámetros de la red de forma efectiva.

Para simplificar y optimizar este proceso, se emplea una técnica conocida como transferencia de aprendizaje. Esta técnica utiliza modelos preentrenados, que han sido previamente entrenados en grandes y variados conjuntos de datos. Estos modelos ya han aprendido a extraer características visuales útiles, como bordes, texturas y formas básicas, que son relevantes para una amplia gama de tareas de visión por computadora.

La transferencia de aprendizaje permite que se aproveche este conocimiento previo, adaptando el modelo preentrenado a nuevas tareas específicas con menos datos y en menos tiempo que si se entrenara un modelo desde cero. En lugar de comenzar desde el principio, se ajustan solo las últimas capas del modelo o se modifican los pesos de las capas intermedias para que se adapten a la nueva tarea. Esto no solo reduce significativamente el tiempo y los recursos necesarios para el entrenamiento, sino que también mejora la precisión al partir de un modelo que ya ha aprendido patrones visuales importantes.

Ejemplos destacados de modelos preentrenados que se utilizan en la transferencia de aprendizaje incluyen VGG16 [58], ResNet [59], Inception [60], DenseNet [61], EfficientNet [62].

Todos estos modelos han sido entrenados en el conjunto de datos *ImageNet* [63], un vasto repositorio de imágenes que contiene más de catorce millones de imágenes etiquetadas en más de veinte mil clases diferentes.

4.3 Decodificador

Después que la CNN procesa las imágenes y produce un vector de características, estas no tienen un significado semántico por sí solas. Para darle sentido a estas características, se utiliza un modelo de procesamiento de lenguaje natural (NLP), que interpreta esta información dentro de un contexto lingüístico. De esta manera, las características visuales se transforman en un formato textual comprensible, facilitando la generación de descripciones que reflejan el contenido de la imagen de manera coherente y precisa.

Un enfoque temprano en el procesamiento del lenguaje natural fue el uso de modelos n-gramas, que se basan en la probabilidad condicional de las palabras en una secuencia. En un modelo n-grama, la probabilidad de una palabra x_t en el paso de tiempo t depende solamente de las $n-1$ palabras anteriores en la secuencia: [64]

$$P(x_t \mid x_{t-1}, x_{t-2}, \ldots, x_{t-n+1})$$

Donde:

- $x_t$es la palabra en el paso de tiempo t.
- $x_{t-1}, x_{t-2}, \ldots, x_{t-n+1}$ son las $n-1$ palabras anteriores.

Si bien los modelos n-gramas fueron útiles en su momento, presentan serias limitaciones. Solo pueden capturar relaciones dentro de una ventana fija de $n-1$ palabras, ignorando cualquier contexto más allá de ese límite. Además, a medida que n aumenta, el número de parámetros del modelo crece exponencialmente, lo que conlleva problemas de eficiencia y sobreajuste, especialmente con vocabularios grandes.

4.3.1 Recurrent Neural Networks (RNN)

Las Redes Neuronales Recurrentes (RNN) se desarrollaron para superar estas limitaciones, las RNN son un tipo de arquitectura de red neuronal diseñada para procesar datos secuenciales, donde cada salida depende no solo de la entrada actual sino también de todas las entradas anteriores, lo que las hace ideales para tareas de generación de texto donde el contexto es crucial.

A diferencia de las redes neuronales *feedforward* o unidireccionales [65], donde la información fluye en una sola dirección, sin retroalimentación o ciclos; las RNN tienen conexiones cíclicas que permiten que la información persista. Esta estructura cíclica introduce un "estado oculto"(h_t), dicha estructura es una representación intermedia que captura la información procesada hasta un determinado punto en una secuencia. Este estado se actualiza en cada paso de la secuencia, permitiendo a la RNN "recordar" entradas anteriores y utilizar esa memoria para influir en las predicciones futuras. De esta manera, el estado oculto actúa como una memoria temporal que resume la información clave de los pasos previos.

La estructura básica de una RNN se compone de tres capas: la capa de entrada, la capa recurrente y la capa de salida.

Capa de Entrada: Recibe y procesa los datos secuenciales de entrada, convirtiéndolos en vectores de características. Cada palabra de una secuencia se puede representar mediante un vector de *embeddings*, que captura su significado semántico.

Un vector de *embeddings* es una representación numérica de una palabra en un espacio vectorial de dimensiones reducidas. En este espacio, palabras con significados similares están representadas por vectores cercanos entre sí, mientras que palabras con significados diferentes están más alejadas. Por ejemplo, la representación de la palabra "fútbol" puede estar cerca de "deporte" y "pelota" en el espacio vectorial, mientras que estará más alejado de términos como "leche" o "vaca".

La creación de estos vectores se basa en un proceso de entrenamiento sobre un corpus de texto grande y diverso. Este corpus proporciona el contexto en el que las palabras aparecen, permitiendo al modelo aprender y capturar las relaciones semánticas y sintácticas entre ellas. Los *embeddings* se entrenan para reflejar estas relaciones contextuales, resultando en representaciones que no solo facilitan el procesamiento computacional, sino que también ofrecen una comprensión más profunda del significado y uso de las palabras. Ejemplos de *embeddings* ya entrenados incluyen *Word2Vec* [18], *GloVe* [19] y *FastText* [66], los cuales se entrenaron en grandes corpus de datos.

Capa Recurrente: Una vez procesados los datos de entrada estos pasan a la capa recurrente, la cual es el núcleo de las RNN y está diseñada para manejar la temporalidad en los datos. A diferencia de los modelos n-gramas, donde la probabilidad de una palabra x_t en el paso de tiempo t se calcula en función de un contexto fijo de las $n-1$ palabras anteriores, las RNN modelan la probabilidad condicional de la palabra x_t utilizando el estado oculto h_{t-1} del paso del tiempo anterior.

$$P(x_t \mid x_{t-1}, x_{t-2}, \ldots, x_{t-n+1}) \approx P(x_t \mid h_{t-1})$$

En cada paso de tiempo, el estado oculto se actualiza mediante una función de activación no lineal que toma en cuenta tanto la entrada actual como el estado oculto anterior:

$$h_t = \sigma(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$

Donde:

- W_{xh} es la matriz de pesos que conecta la entrada x_t al estado oculto h_t.
- W_{hh} es la matriz de pesos que conecta el estado oculto h_{t-1}al estado oculto h_t.
- b_h es el vector de sesgo asociado al estado oculto.

Capa de Salida: Por último, la capa de salida produce la salida basada en el estado oculto actual. La probabilidad de la palabra x_t se obtiene a partir de h_t, utilizando una función de activación (como *softmax*) que genera una distribución de probabilidad sobre todas las posibles palabras del vocabulario:

$$P(x_t \mid h_t) = f_s(W_{hy}h_t + b_y)$$

Donde W_{hy} es la matriz de pesos de salida, b_y es el sesgo de salida, y f_s es la función softmax que convierte el vector de activaciones en una distribución de probabilidad.

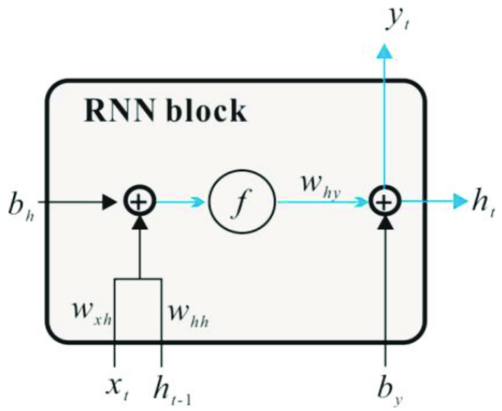

Ilustración 6: Esquema de funcionamiento de una RNN simple.

Sin embargo, las RNN tradicionales enfrentan limitaciones cuando se trata de capturar dependencias a largo plazo debido al problema del desvanecimiento del gradiente [67]. Este problema ocurre cuando los gradientes que se propagan hacia atrás a través de muchas capas o pasos de tiempo se vuelven extremadamente pequeños, lo que impide que la red ajuste eficazmente los pesos relacionados con las entradas tempranas en la secuencia. Como resultado, la red tiene dificultades para aprender y retener información relevante a lo largo de secuencias largas.

4.3.2 Long Short-Term Memory networks (LSTM)

Para abordar el problema del desvanecimiento del gradiente y mejorar el manejo de dependencias a largo plazo, se han desarrollado diversas variantes de las RNN, en este trabajo vamos a tratar las *Long Short-Term Memory networks* (LSTM). Las LSTM son una evolución de las RNN diseñadas para superar estos desafíos. La clave radica en su capacidad para mantener la información durante períodos prolongados.

La principal innovación de las LSTM es la introducción de celdas de memoria y tres tipos de puertas controladoras que regulan el flujo de información. Esto permite a las LSTM "almacenar, olvidar o acceder" a la información de manera controlada, abordando directamente el problema del desvanecimiento del gradiente. La celda de memoria es el núcleo, encargada de almacenar y actualizar la información relevante a lo largo del tiempo. Esta se actualiza en cada paso de tiempo en función tanto de la nueva información como la información almacenada previamente. [67]

Cada una de las tres puertas que controlan el flujo de información está diseñada con una capa sigmoidea, que actúa como un filtro determinando qué fracción de la información debe pasar a la siguiente etapa. Este filtrado se realiza a través de una operación de multiplicación punto a punto, la cual ajusta con precisión los valores de la información a ser retenida o descartada.

La puerta de entrada (*input gate*) controla qué parte de la nueva información que se está procesando debe ser incorporada a la celda de memoria, asegurando que solo los datos relevantes sean incorporados. Esta se define mediante:

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

La salida de la puerta de entrada i_t indica la fracción de la nueva información que se debe considerar. Un valor cercano a 1 indica que la nueva información es completamente relevante, mientras que un valor cercano a 0 que se debe ignorar. Además, se calcula el candidato a celda $\hat{C}_t$, que es la nueva información propuesta para la celda de memoria:

$$\hat{C}_t = tanh(W_{\hat{C}} \cdot [x_t, h_{t-1}] + b_{\hat{C}})$$

Por otro lado, la puerta de olvido (*forget gate*) se encarga de eliminar la información innecesaria o irrelevante de la celda de memoria, manteniendo así la eficiencia de la red en cada paso de tiempo. La puerta de olvido se define mediante la siguiente fórmula:

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

La salida de la puerta de olvido $f_t$determina qué fracción de la información contenida en la celda de memoria anterior C_{t-1} debe ser mantenida. Un valor cercano a 1 significa que la información previa debe ser retenida casi en su totalidad, mientras que un valor cercano a 0 indica que gran parte de la información debe ser olvidada.

Una vez obtenidas las salidas de ambas puertas la celda de memoria C_t se actualiza combinando la información antigua que se decide conservar mediante la puerta de

olvido f_t, con la nueva información, que se decide agregar mediante la puerta de entrada i_t y el candidato a celda $\hat{C}_t$.

$$C_t = f_t \circ C_{t-1} + i_t \circ \hat{C}_t$$

Finalmente, la puerta de salida (*output gate*) determina qué información contenida en la celda de memoria debe ser utilizada para producir la salida en el paso de tiempo actual. Esta puerta asegura que solo la información relevante sea reflejada en la salida de la red. La puerta de salida se define mediante:

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

Un valor cercano a 1 en significa que la información presente en C_t se utilizará casi en su totalidad para producir la salida, mientras que un valor cercano a 0 indica que gran parte de la información será suprimida en la salida, permitiendo que sólo una fracción limitada influya en el estado oculto h_t.

La salida de la puerta de salida o_t modula la celda de memoria actual C_t a través de la función tangente hiperbólica para determinar el estado oculto h_t en el paso de tiempo t :

$$h_t = o_t \circ tanh(C_t)$$

Aquí, o_t actúa como un filtro que ajusta la magnitud de la información que se extrae de C_t, mientras que la función $tanh(C_t)$ asegura que los valores se mantengan en un rango adecuado, garantizando que el estado oculto h_t refleje de manera precisa y controlada la información relevante almacenada en la celda de memoria.

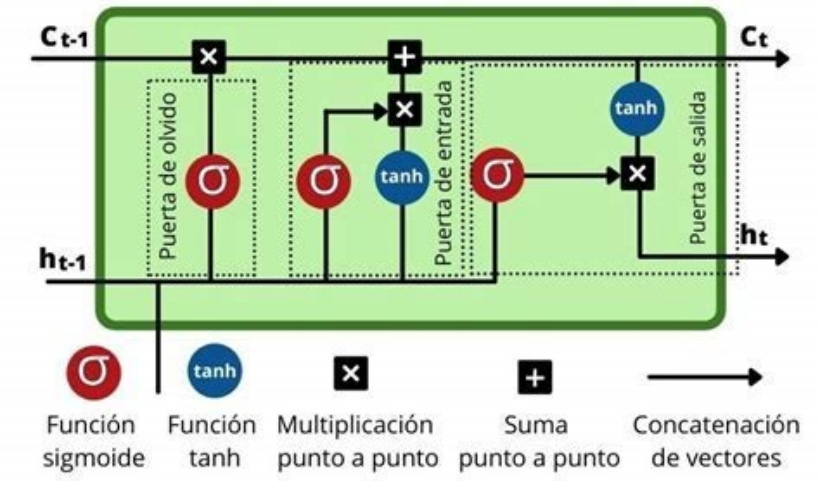

Ilustración 7: Esquema de funcionamiento de una LSTM

El "carrusel de error constante" es otro mecanismo clave en las LSTM que permite que los gradientes fluyan sin disminuir significativamente a medida que se propagan hacia atrás en el tiempo. Esto se logra mediante la estructura interna de la celda de memoria, que preserva las señales de error y permite mantener información relevante durante secuencias largas. Al combinar este mecanismo con las puertas de control y la celda de memoria, se logra mitigar el problema del desvanecimiento del gradiente.

En el contexto de este Trabajo de Fin de Grado, las LSTM reciben como entrada inicial el vector de características visuales de la imagen obtenido mediante la CNN, representadas como vectores de alto nivel. Este vector sirve únicamente para inicializar la LSTM, mientras que la generación de palabras se realiza a través de los *embeddings* de palabras, los cuales utiliza para producir descripciones secuenciales a partir de un marcador de inicio `<start>` hasta el marcador de fin `<end>`.

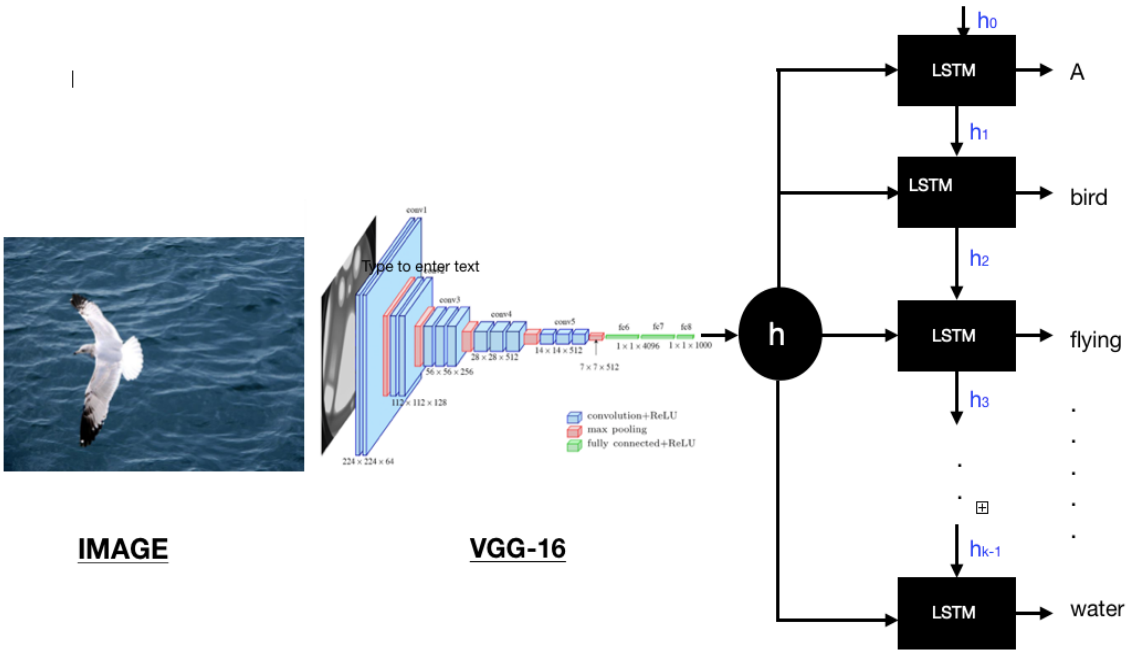

Ilustración 8: Modelo codificador-decodificador *(CNN+LSTM). [68]*

La capacidad de las LSTM para mantener y gestionar dependencias a largo plazo resulta especialmente valiosa en este contexto, ya que permite que la descripción generada capture no solo detalles inmediatos, sino también aspectos más generales y contextuales de la imagen.

El problema de este enfoque se debe a que, al generar cada palabra en la descripción, el modelo normalmente se enfoca solo en una parte de la imagen, sin captar la totalidad de su contenido. Esto se debe a que, al generar cada palabra, el modelo tiende a usar una representación general de la imagen que puede no ser suficiente para capturar detalles complejos y específicos de todas las áreas relevantes de la imagen. Para superar esta limitación, se introducen los mecanismos de atención.

4.3.3 Mecanismos de atención

El primer mecanismo de atención, también conocido como Bahdanau attention, fue introducido en 2014 en el artículo "*Neural Machine Translation by Jointly Learning to Align and Translate*" [69]. Este innovador enfoque permite al modelo asignar diferentes pesos de importancia a las distintas regiones de la entrada simultáneamente, lo que facilita una consideración más detallada de la información en cada paso del proceso.

Los mecanismos de atención permiten que el modelo se enfoque en diferentes partes de la imagen durante el proceso de generación de texto. Inspirados en el patrón de atención humana, donde el ojo humano se centra en las características salientes de una imagen, los métodos basados en atención replican este enfoque en el contexto de redes neuronales. En lugar de tratar la imagen como una entidad única, el mecanismo de atención divide la imagen en múltiples regiones.

Cuando el modelo de atención está generando una nueva palabra, se centra en la región específica de la imagen que es más relevante para esa palabra en particular. Esto significa que, en lugar de utilizar la representación global de la imagen para todo el proceso de generación de palabras, el modelo adapta su enfoque a las partes más importantes en cada etapa.

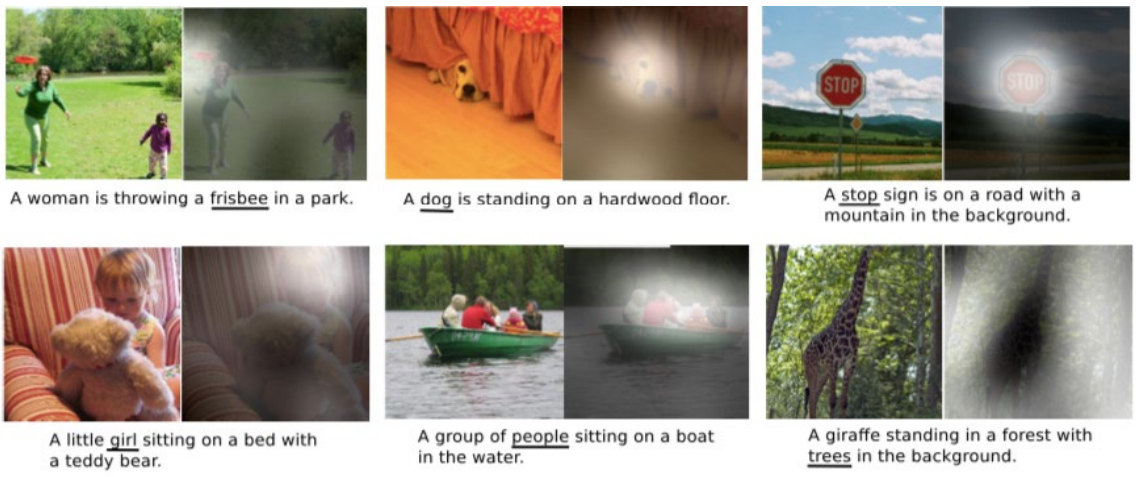

Ilustración 9: Ejemplos de descripciones con atención visual. [70]

Con la incorporación de una capa de atención, si el modelo ha generado ya varias palabras, el estado oculto de la LSTM se utiliza para seleccionar la parte de la imagen que es más relevante en ese momento. Esta representación filtrada de la imagen, que solo conserva las partes pertinentes, se pasa al LSTM, que luego predice la siguiente palabra y actualiza su estado oculto en consecuencia. Durante el entrenamiento, el modelo aprende a identificar las regiones más relevantes de la imagen. En este proceso, una LSTM no solo utiliza la salida generada y el estado oculto previo, sino también un vector de contexto c, que refleja la información más significativa de la imagen en cada paso. [68]

La manera en que este vector de contexto se calcula y utiliza puede variar dependiendo del tipo de mecanismo de atención, pero su propósito fundamental es siempre el mismo: proporcionar al decodificador información relevante sobre la entrada (en este caso, una imagen o secuencia de características visuales) para generar la próxima palabra o salida.

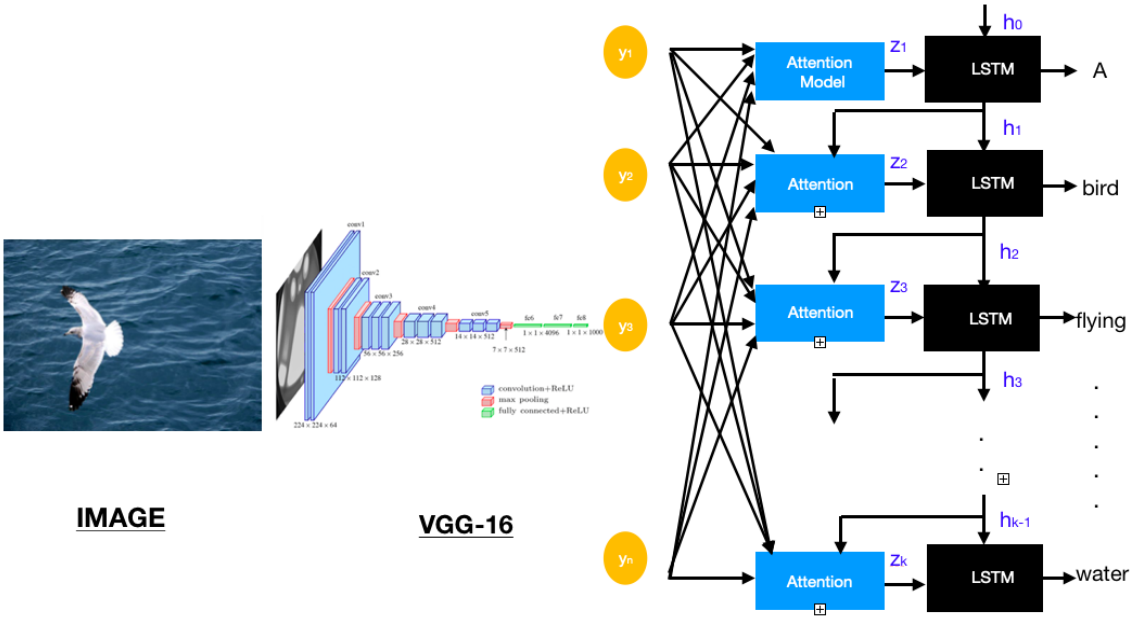

Ilustración 10: Modelo codificador-decodificador *con atención. [68]*

Tipos de Mecanismos de Atención:

- **Soft Attention:** [23] Este mecanismo permite al modelo asignar diferentes pesos de importancia a todas las regiones de la entrada simultáneamente, permitiendo que el modelo considere toda la información en cada paso, pero con diferentes grados de énfasis.

- **Hard Attention:** [23] Selecciona una única región de interés para procesar en cada paso, en lugar de considerar toda la entrada simultáneamente.

- **Local Attention:** [71] Se enfoca en una ventana específica que se desplaza a lo largo de la imagen, ajustando continuamente el campo de visión.

- **Global Attention:** [71] Evalúa toda la entrada para determinar la relevancia de cada parte en la generación de cada palabra, ofreciendo una vista completa.

- **Dot-Product Attention:** [72] Calcula la similitud entre consultas y claves usando el producto escalar para determinar la atención a diferentes partes de la entrada. En el contexto de generación de descripciones de imágenes, las consultas son las representaciones del estado del decodificador, mientras que las claves son las representaciones vectoriales de diferentes regiones de la imagen extraídas por la CNN.

- **Multi-Head Attention:** [72] Expande el concepto de *dot-product attention* dividiendo las consultas, claves y valores en múltiples subespacios. Cada "cabeza" de atención realiza su propia operación de atención y los resultados se combinan.

- **Bottom-Up Attention:** [73] Utiliza una CNN para identificar regiones de interés en una imagen, dirigiendo la atención a las áreas más significativas.

Tradicionalmente, estos mecanismos de atención se implementaban en combinación con redes neuronales recurrentes, como las LSTM, para aprovechar la capacidad de las RNNs para manejar datos secuenciales y la capacidad de los mecanismos de atención para enfocar el modelo en las partes relevantes de la entrada.

Sin embargo, un cambio significativo ocurrió en 2017 con la publicación del artículo "*Attention Is All You Need"* por Vaswani et al. [72]. Este trabajo revolucionó el campo al sugerir que las redes neuronales recurrentes podían ser prescindibles para tareas de procesamiento secuencial cuando se utilizaban mecanismos de atención de manera eficiente.

El artículo presentó los modelos *Transformers*, que basan su arquitectura exclusivamente en mecanismos de atención, eliminando la necesidad de redes recurrentes. Los *Transformers* demostraron que los mecanismos de atención por sí solos pueden ser suficientes para capturar dependencias complejas y relaciones a largo plazo en los datos, ofreciendo mejoras significativas en el rendimiento y eficiencia de los modelos. Este enfoque ha llevado a avances importantes en el procesamiento del lenguaje natural y la generación de descripciones de imágenes, estableciendo un nuevo estándar en el estado del arte de la inteligencia artificial.

4.4 Transformers

Los modelos *Transformers*, también se basan en la arquitectura Codificador-Decodificador. En ellos, el codificador convierte la entrada en una serie de representaciones internas utilizando mecanismos de atención, y el decodificador toma estas representaciones para producir la salida. Así, los *Transformers* aplican la misma estructura básica, pero con una implementación optimizada y sin depender de las RNNs tradicionales.

El corazón del modelo *Transformer* es el mecanismo de atención propia o *self-attention*. A diferencia de las RNNs, que procesan secuencias de datos de forma secuencial, permite que el modelo evalúe la relevancia de cada parte de la secuencia con respecto a todas las demás partes simultáneamente.

El mecanismo *self-attention* se calcula utilizando tres matrices: Q (*queries*), K (*keys*) y V (*values*). Estas matrices se derivan de la entrada mediante proyecciones lineales. El cálculo se realiza utilizando el producto escalar escalado:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}}) \cdot V$$

Donde QK^T representa el producto escalar entre las consultas y las claves, escalado por la raíz cuadrada de la dimensión de las claves d_k. Luego, se aplica la función *softmax* para obtener una distribución de probabilidad, que se utiliza para ponderar V.

Para potenciar aún más la capacidad del modelo para capturar relaciones complejas en los datos, el *Transformer* emplea multi-head attention. En lugar de aplicar el mecanismo de *self-attention* una sola vez, lo ejecuta en paralelo múltiples veces, utilizando distintas proyecciones lineales para generar varias versiones de Q, K y V. Los resultados de estas múltiples "cabezas" de atención se combinan al final, lo que permite al modelo capturar relaciones en diferentes subespacios.

Mediante este proceso el modelo asigna diferentes grados de importancia a distintas partes de la secuencia de entrada, permitiendo al modelo enfocarse en las partes relevantes de la secuencia, independientemente de su posición, lo que facilita la captura de dependencias a largo plazo y la identificación de patrones complejos en los datos.

El *Transformer* está compuesto por dos bloques principales: el codificador y el decodificador. Cada uno de los bloques está formado por múltiples capas, y la estructura general del modelo se detalla en el diagrama del artículo "*Attention Is All You Need* ".

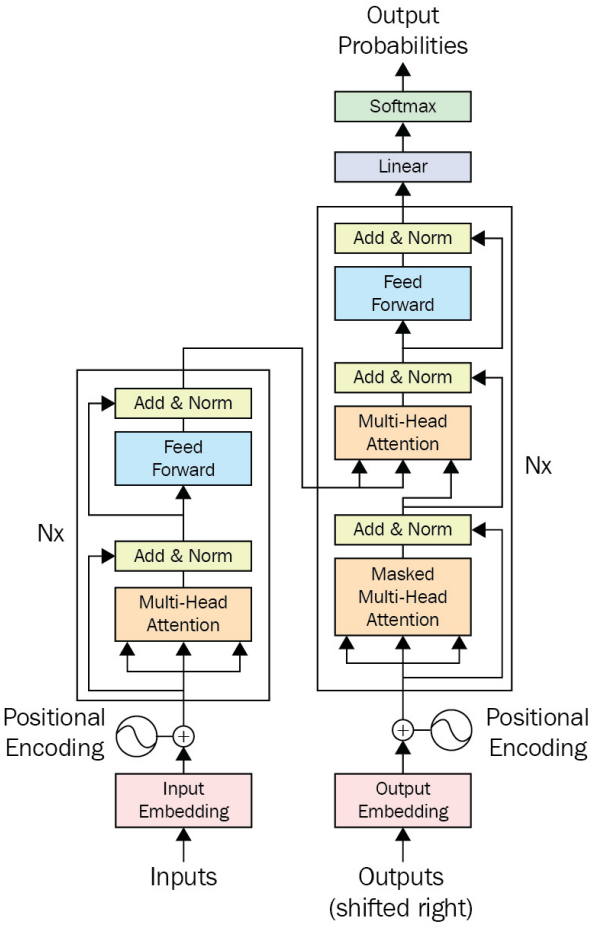

Ilustración 11: Diagrama de un modelo Transformer. [71]

Como se observa en la ilustración 12, en primer lugar, el modelo necesita una manera de incorporar información sobre el orden de las palabras en la secuencia, para ello hace uso del *positional encoding*. Este componente lo que hace es añadir una representación vectorial a cada token de la entrada para codificar su posición en la secuencia. Utiliza funciones seno y coseno de diferentes frecuencias para generar estas representaciones, que se suman a los *embeddings* de los tokens, asegurando que la posición de cada token en la secuencia se tenga en cuenta de manera que la red pueda aprender patrones basados en el orden.

Codificador:

El codificador está compuesto por una serie de N capas idénticas, donde cada una sigue un flujo de procesamiento definido. En primer lugar, la entrada pasa por un mecanismo de multi-head attention, el cual, como ya se ha explicado, aplica *self-attention* en paralelo múltiples veces permitiendo que cada posición en la secuencia de entrada interactúe con todas las demás posiciones, capturando así relaciones tanto locales como globales.

Una vez que la entrada ha pasado por el mecanismo de atención propia, pasa por una red neuronal *feed-forward* completamente conectada. Esta red aplica una serie de transformaciones no lineales, lo que permite al modelo identificar y capturar patrones más complejos en la secuencia de datos.

Para asegurar la estabilidad del entrenamiento y prevenir el sobreajuste, se incluyen técnicas adicionales como la normalización por capas, conexiones residuales y *dropout*. La normalización por capas ayuda a estabilizar el entrenamiento al normalizar las activaciones, las conexiones residuales permiten el flujo de gradientes más eficiente y el *dropout* actúa como una forma de regularización para evitar el sobreajuste.

Decodificador:

El decodificador tiene una estructura similar a la del codificador, con el mismo número de capas. Sin embargo, cada capa incluye una capa adicional de Codificador-Decodificador *attention*. Este mecanismo permite que el modelo enfoque su atención en partes específicas de la secuencia de entrada mientras genera la secuencia de salida.

El decodificador opera de manera autorregresiva, generando la secuencia de salida un token a la vez. En cada paso del proceso, el modelo utiliza los tokens previamente generados como entrada para predecir el siguiente token en la secuencia, ajustando las predicciones en función de la información de los tokens ya producidos.

Una vez que la información ha pasado a través de las diferentes capas, se procesa a través de una capa lineal que proyecta las representaciones internas en un espacio de dimensionalidad igual al tamaño del vocabulario. Posteriormente, se aplica una función *softmax* para convertir estas puntuaciones lineales en probabilidades, permitiendo al modelo seleccionar el token más probable como el siguiente en la secuencia generada.

4.4.1 ViT (Vision *Transformer*)

Además del impacto significativo que los *Transformers* han tenido en el procesamiento del lenguaje natural, su arquitectura ha sido adaptada exitosamente para abordar problemas en el campo de la visión por computadora, dando lugar a una innovadora variante conocida como *Vision Transformers* (ViT). Mientras que las Redes Neuronales Convolucionales (CNNs) han sido tradicionalmente la elección principal para tareas de visión por computadora debido a su capacidad para capturar patrones locales en imágenes, los ViT introducen un enfoque completamente diferente basado en la atención, permitiendo capturar relaciones globales en las imágenes desde las primeras capas del modelo.

Los *Vision Transformers* (ViT), introducidos por Dosovitskiy et al. en 2020 [20], adaptan directamente la arquitectura del *Transformer* para trabajar con datos visuales. En lugar de procesar secuencias de texto, los ViT operan sobre imágenes dividiéndolas en parches más pequeños, que se tratan de manera análoga a los "tokens" en tareas de lenguaje natural. Por ejemplo, una imagen de 224x224 píxeles puede dividirse en parches de 16x16 píxeles, resultando en 196 parches. Cada parche se aplana y se proyecta en un espacio de características de dimensión fija mediante una capa de *embedding*. Este proceso convierte cada parche en un vector que el modelo puede procesar.

Para que el modelo comprenda las relaciones espaciales entre los parches de la imagen, los ViT utilizan *positional encodings*, similares a los usados en los *Transformers* de lenguaje. Estas codificaciones se añaden a los *embeddings* de los parches para incorporar información sobre la posición de cada parche dentro de la imagen, esencial para preservar la estructura espacial. [74]

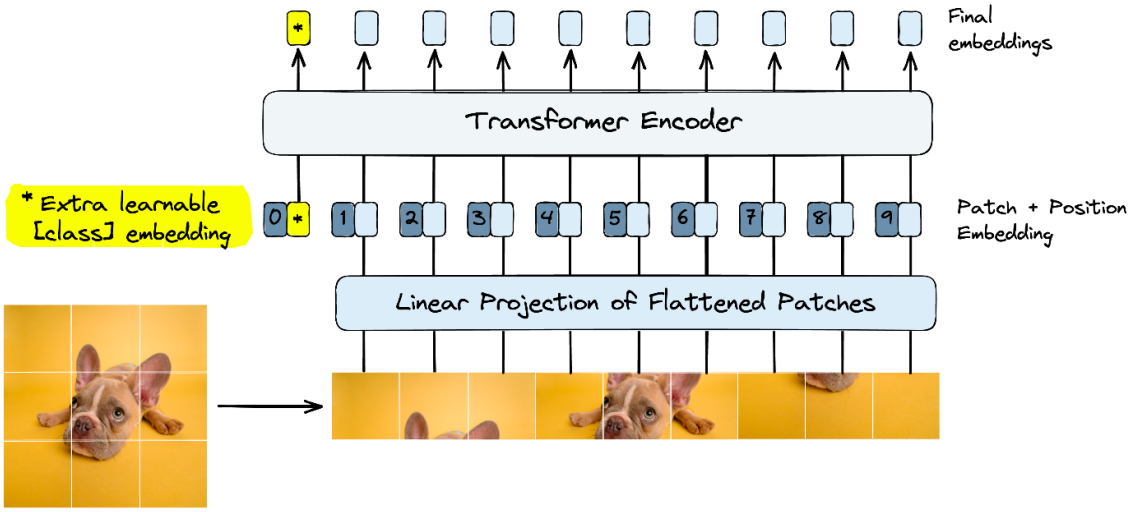

Ilustración 12: Funcionamiento Vision Transformer. [74]

El componente central, al igual que en el *Transformer* tradicional, es el mecanismo de *self-attention*. Este mecanismo permite que cada parche de la imagen "preste atención" a todos los demás parches simultáneamente, determinando cuáles son más relevantes para la tarea en cuestión. Esto proporciona al modelo una comprensión global de la imagen desde las primeras etapas del procesamiento, en contraste con las

CNNs, que construyen esta comprensión de manera más gradual a través de múltiples capas convolucionales.

En los ViT, las imágenes se dividen en parches que se transforman en secuencias de vectores. Estos vectores, junto con un token especial de clasificación, [CLS], se introducen en el modelo. El token [CLS] se añade a la secuencia de parches y, tras pasar por múltiples capas de *self-attention* y transformaciones de red neuronal *feed-forward*, captura la información global agregada de toda la imagen. Este token final es utilizado para realizar la predicción final, como la clasificación de imágenes o la extracción de características relevantes para otras tareas.

Una de las principales ventajas de los *Vision Transformers* es su capacidad para capturar relaciones a largo plazo y dependencias globales en las imágenes. Sin embargo, esta flexibilidad y capacidad para capturar dependencias globales también presentan desafíos, ya que requieren de grandes cantidades de datos para entrenarse de manera efectiva y evitar problemas de sobreajuste.

En esta tarea, se emplearán ViT para extraer características visuales de las imágenes y, posteriormente, se empleará el modelo GPT-2 para generar descripciones textuales coherentes y detalladas basadas en estas características. El ViT se encargará de analizar la imagen, capturando patrones, formas y texturas a través de su mecanismo de atención, mientras que GPT-2 utilizará la información extraída para construir una descripción en lenguaje natural que refleje con precisión el contenido visual de la imagen.

Los Vision *Transformers*, al igual que otros modelos basados en *Transformers*, se benefician de un entrenamiento extensivo en grandes conjuntos de datos visuales para aprender representaciones generales de imágenes. Este preentrenamiento se realiza sobre grandes corpus de imágenes para que el modelo pueda aprender a identificar patrones visuales básicos y complejos. Posteriormente, se realiza el *fine-tuning* para adaptar el modelo a tareas específicas. En este trabajo, el ViT se entrena adicionalmente con el conjunto de datos Flickr8K para la generación de descripciones de imágenes. Este *fine-tunning* adapta los parámetros del modelo a las características del nuevo conjunto de datos, mejorando su capacidad para generar descripciones precisas y coherentes.

4.4.2 GPT-2 (Generative Pre-trained *Transformer* 2)

Se trata de un modelo de lenguaje desarrollado por OpenAI [75] que ha marcado un hito en el procesamiento del lenguaje natural debido a su capacidad para generar texto coherente y contextualmente relevante. GPT-2 es conocido por su tamaño, con hasta 1.5 mil millones de parámetros en su versión más grande.

Se trata de un modelo de lenguaje basado en la arquitectura *Transformer*, se ha adaptado para tareas de generación automática de descripciones de imágenes mediante el uso de características visuales extraídas previamente. En este contexto, no

solo genera texto basado en secuencias de palabras, sino que también utiliza la información visual para producir descripciones detalladas y coherentes.

El proceso de entrenamiento de GPT-2 incluye un preentrenamiento extenso en grandes corpus de texto, que abarcan una amplia variedad de temas y estilos lingüísticos, lo que permite al modelo aprender una comprensión profunda del lenguaje y sus estructuras. Durante este proceso, aprende a predecir la siguiente palabra en una secuencia de texto dada su historia previa, lo que le permite captar patrones lingüísticos complejos y relaciones contextuales entre palabras. Este entrenamiento a gran escala le proporciona al modelo una base sólida en el lenguaje. Una vez completado el preentrenamiento, GPT-2 puede ser ajustado para tareas específicas mediante un proceso denominado *fine-tuning*. En esta etapa, el modelo se entrena adicionalmente con un conjunto de datos más específico y relevante para la tarea deseada, como la generación de descripciones de imágenes. El *fine-tuning* ajusta los parámetros del modelo para que se adapte mejor a las características del nuevo conjunto de datos. Por ejemplo, en este trabajo el *fine-tuning* se realiza utilizando el conjunto de datos Flickr8K.

En la tarea específica de la generación de descripciones, las características visuales de la imagen extraídas por el *Vision Transformer* se utilizan como entrada para el modelo. Estas características se convierten en vectores que representan los elementos visuales y patrones complejos identificados en la imagen. La arquitectura de atención de GPT-2 permite que el modelo integre y contextualice esta información visual, generando descripciones que capturan tanto los detalles visibles como el contexto general de la imagen.

Durante la fase de decodificación o inferencia, el modelo utiliza su capacidad de atención para generar texto basado en las características visuales proporcionadas. Este proceso permite crear descripciones ricas y contextualmente apropiadas que van más allá de la simple enumeración de objetos, incorporando relaciones espaciales, actividades y escenarios presentes en la imagen. Gracias a su entrenamiento previo y su arquitectura avanzada, GPT-2 puede generar narrativas detalladas y comprensibles que reflejan de manera precisa el contenido visual.

5. Recursos utilizados

5.1 Entorno y lenguaje de programación

Para el desarrollo de las diversas implementaciones de programas generadores de descripciones, se han empleado diversas redes neuronales y metodologías avanzadas, previamente explicadas. Debido a la insuficiente capacidad de memoria RAM de mi ordenador (8GB) para gestionar las exigencias del proyecto, se ha optado por utilizar *Google Colab Pro*[4]. Esta plataforma en la nube proporciona acceso a recursos computacionales de alto rendimiento, como las TPU (*Tensor Processing Unit*) y las GPU (*Graphics Processing Unit*).

Las TPUs son particularmente efectivas para tareas de *deep learning* debido a su arquitectura especializada en la realización de operaciones matemáticas intensivas en paralelo. Este diseño permite acelerar significativamente el entrenamiento de modelos de redes neuronales al optimizar el procesamiento de grandes volúmenes de datos y la ejecución de operaciones tensoriales, que son fundamentales para el desarrollo de modelos complejos. En este proyecto, se ha utilizado una TPU con una capacidad de memoria RAM de 334.6 GB, lo que ha facilitado un procesamiento más ágil y eficiente, mejorando notablemente el rendimiento del entrenamiento del modelo.

Además, el uso de GPU ha sido fundamental para acelerar tanto el entrenamiento como la inferencia de los modelos de *deep learning*. Las GPUs son especialmente útiles para realizar cálculos paralelos en redes neuronales profundas, reduciendo el tiempo requerido para completar las tareas de entrenamiento y ajuste. En este proyecto, se utilizó una GPU para implementar el modelo ViT + GPT-2, ya que este requería el uso de CUDA para aprovechar la aceleración por hardware en el entrenamiento del modelo. CUDA permite que las GPU realicen operaciones de cómputo intensivas de manera más eficiente, lo que es esencial para manejar modelos tan grandes y complejos.

Google Colab Pro permite la ejecución de código en un entorno de notebooks Jupyter, utilizando el lenguaje de programación Python, ampliamente empleado en el desarrollo de modelos de aprendizaje automático y redes neuronales.

En este proyecto, se ha optado por Python debido a la amplia disponibilidad de bibliotecas y frameworks diseñados específicamente para aplicaciones de *deep learning*.

[4] https://colab.research.google.com/

5.2 *Dataset*

Para la realización del proyecto, se consideraron varias opciones de *datasets*, como COCO, *Flickr30K* y *Flickr8K*. Finalmente, como se comentó anteriormente se optó por utilizar *Flickr8K* debido a su equilibrio entre tamaño, diversidad y la calidad de las descripciones.

El *dataset* se descargó desde Kaggle [76] e incluye seis archivos de texto, así como una carpeta llamada 'Images', donde se encuentran todas las imágenes.

En cuanto a los archivos de texto, '*Flickr8K*.token.txt' contiene las descripciones asociadas a cada imagen del conjunto de datos. En este archivo, cada línea incluye un identificador único de la descripción, que sigue el formato "id de imagen # id de descripción". Este identificador vincula cada descripción con una imagen específica en el conjunto de datos.

Los archivos `Flickr_8k.trainImages.txt`, `Flickr_8k.devImages.txt` y `Flickr_8k.testImages.txt` contienen listas de imágenes clasificadas en tres conjuntos diferentes: 6000 imágenes para el entrenamiento, 1000 para la validación y 1000 para la prueba. Estos archivos enumeran las imágenes que se utilizarán para cada fase del experimento.

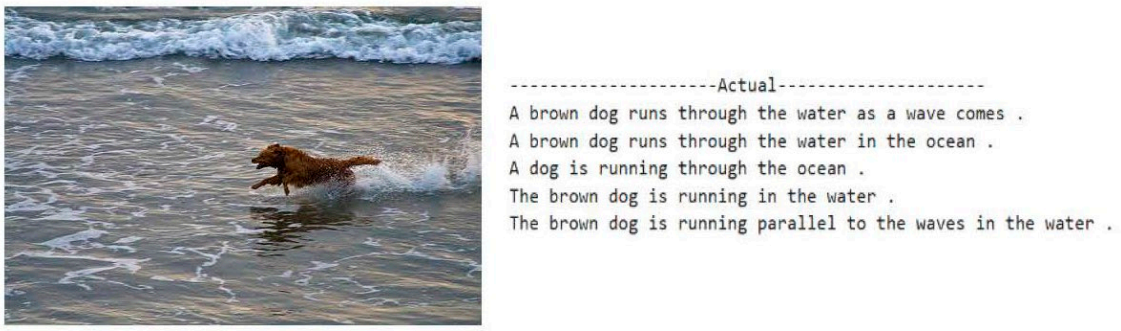

Ilustración 13: Imagen junto con sus descripciones referencia.

6. Marco experimental

En este Trabajo de Fin de Grado, se han implementado y evaluado dos arquitecturas de modelos para la tarea de descripción automática de imágenes, con el fin de demostrar su evolución y analizar cómo cada una ha contribuido a mejorar el rendimiento en esta área. Las arquitecturas consideradas son: CNN + LSTM y una basada en *Transformers*.

El objetivo de comparar estas dos arquitecturas es ilustrar cómo cada una ha supuesto un avance respecto a la otra, mostrando claramente el progreso en términos de precisión y coherencia en la generación de descripciones de imágenes. Esta comparación permite evidenciar cómo las técnicas más avanzadas, como los *Transformers*, han logrado un rendimiento superior en la tarea de descripción automática de imágenes, mejorando significativamente la calidad y relevancia de las descripciones generadas.

6.1 Arquitectura 1: Codificador-Decodificador (CNN+LSTM)

La primera arquitectura implementada en este Trabajo de Fin de Grado es el modelo Codificador-Decodificador basado en una combinación de Redes Neuronales Convolucionales (CNN) y Long Short-Term Memory (LSTM). En esta configuración, como ya se ha explicado, la CNN se encarga de extraer características visuales de las imágenes, transformándolas en un vector de características que resume la información visual importante. Este vector es luego alimentado a una LSTM, que actúa como el decodificador, generando la descripción textual de la imagen.

6.1.1 Bibliotecas

Las bibliotecas utilizadas en esta primera arquitectura son:

- **Numpy** es una biblioteca enfocada en el cálculo numérico y el análisis de datos. Utilizada para almacenar y procesar características visuales y secuencias de texto en arrays multidimensionales, realizando operaciones matemáticas y estadísticas rápidas.
- **Pydot** y **Graphviz**, utilizadas para visualizar y generar gráficos de modelos de redes neuronales.
- **Pickle**, empleada para guardar y cargar las características visuales obtenidas mediante las CNNs.

- **Pandas**, facilita la carga, limpieza y transformación de datos a través de su estructura de DataFrame.
- **Os**, para interactuar con el sistema operativo, como manejar rutas de archivos, listar directorios, y verificar la existencia de archivos.
- **String**, para manipular cadenas de texto.
- **Tqdm**, proporciona barras de progreso en las celdas de notebooks para un seguimiento visual del progreso.
- **Nltk.translate.bleu_score**, se utiliza para calcular la métrica BLEU, que evalúa la calidad de texto generado comparándolo con un conjunto de referencias.
- **Matplotlib**, se utiliza para cargar y visualizarlas imágenes.
- **Random,** para seleccionar imágenes aleatorias para la generación de descripciones.

Por último, es importante destacar dos bibliotecas clave en el ámbito del *deep learning*: **TensorFlow**[5] y **Keras**[6].

Ambas se utilizan en conjunto para construir, entrenar, y evaluar modelos de aprendizaje profundo. TensorFlow es una biblioteca de código abierto desarrollada por Google para la computación numérica y el aprendizaje automático. Su propósito principal es facilitar el desarrollo y la implementación de modelos de *machine learning* y *deep learning* a gran escala. Keras es una biblioteca de alto nivel que se utiliza como una interfaz simplificada sobre TensorFlow. Su objetivo principal es hacer que el diseño, entrenamiento y evaluación de modelos de *deep learning* sea más accesible y menos propenso a errores.

TensorFlow maneja la computación subyacente, optimización, y entrenamiento distribuido, mientras que Keras proporciona una interfaz simplificada para definir, compilar, y entrenar los modelos de *deep learning*.

6.1.2 Codificación de imágenes

Para el proceso de extracción de características visuales de las imágenes, se ha elegido el modelo preentrenado VGG16 por su eficacia en la obtención de características representativas con una estructura relativamente simple y sin requerir un elevado poder computacional. En lugar de usar el modelo completo para la clasificación de imágenes, se ajusta para que la salida corresponda al vector de 4096 características de la penúltima capa densa, transformándolo en un eficiente extractor de características.

El modelo VGG16 está compuesto por trece capas convolucionales, cinco capas de agrupamiento (*pooling*) y dos capas totalmente conectadas. Su arquitectura se basa en la repetición de bloques de capas convolucionales seguidas de capas de agrupamiento max-*pooling*, lo que permite capturar información detallada en diferentes niveles de abstracción. Las últimas capas, que son completamente conectadas, combinan estas

[5] https://www.tensorflow.org/?hl=es
[6] https://keras.io/

características para formar una representación compacta y rica en información, que servirá más adelante para obtener las descripciones.

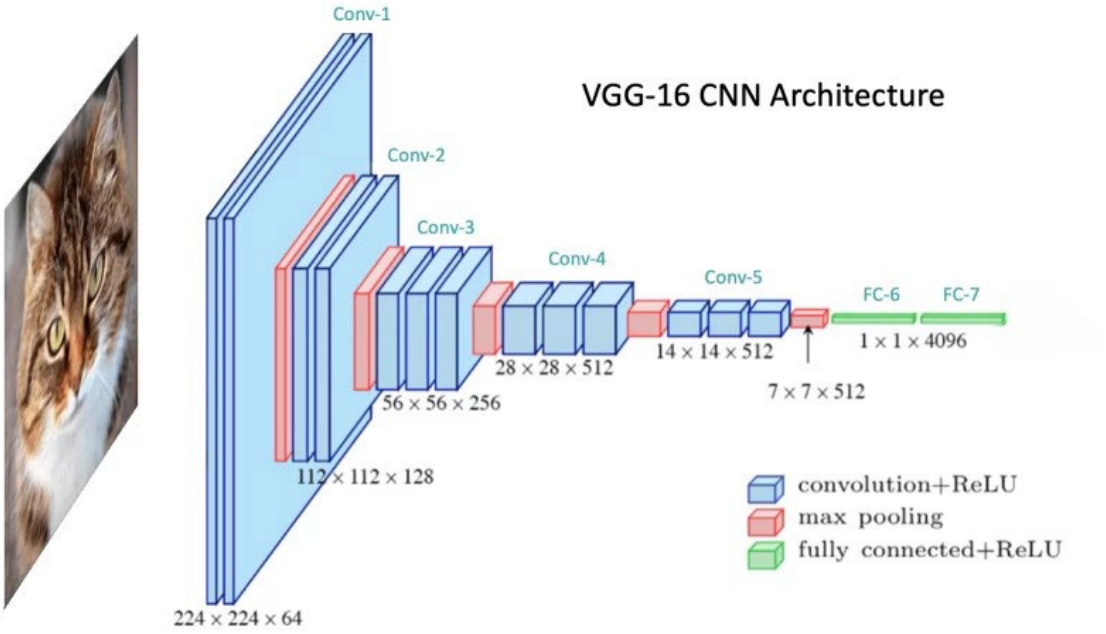

Ilustración 14: Estructura VGG16. [77]

Cada imagen del conjunto de datos se carga y redimensiona a 224x224 píxeles, el tamaño requerido por VGG16. Luego, la imagen se convierte en un array *NumPy* y se ajusta para tener un lote de tamaño 1, lo que es necesario para que el modelo procese las imágenes en lotes. Las imágenes se preprocesan para cumplir con los requisitos del modelo, incluyendo la normalización de los valores de los píxeles. Tras el preprocesamiento, las imágenes se introducen en el modelo para obtener un vector de características. Estos vectores se almacenan en un diccionario, donde cada entrada está asociada con un identificador único de imagen.

Mediante la librería *pickle* el diccionario de características se guarda en un archivo para su posterior uso sin necesidad de volver a ejecutarlo.

6.1.3 Decodificación de descripciones

Preprocesado de texto

Para lograr que el modelo LSTM funcione de manera eficiente, es esencial que las descripciones de texto estén en un formato adecuado, lo que requiere un preprocesamiento previo. Para ello, se comienza cargando el archivo *Flickr8K.token.txt*, que contiene las descripciones asociadas a cada imagen. Se define una función denominada *load_captions()* con el objetivo de almacenar las descripciones en un diccionario. En este diccionario, las claves corresponden a los identificadores de las imágenes, y los valores son listas que contienen las cinco descripciones asociadas a

cada imagen. Esta estructura permite una asociación clara y accesible entre las imágenes y sus descripciones.

Una vez creado, la función *clean_captions()* recibe dicho diccionario como entrada y se encarga de limpiar las descripciones. En primer lugar, convierte todas las descripciones a minúsculas para asegurar la uniformidad en el análisis de texto. Luego, elimina cualquier puntuación que pueda interferir con la tokenización y el análisis posterior. A continuación, la función limpia el texto eliminando espacios en exceso y filtra palabras de un solo carácter o que contengan números, ya que estas suelen ser irrelevantes para el modelo de generación de texto. Finalmente, la función añade tokens especiales de inicio (`'startseq'`) y fin (`'endseq'`) a cada descripción, los cuales son utilizados por el modelo para marcar el comienzo y el final de cada secuencia generada durante el entrenamiento.

Tras la limpieza de las descripciones, se procede a crear un vocabulario que incluya todas las palabras presentes en las descripciones. La función *to_vocabulary()* extrae todas las palabras únicas de las descripciones limpias, creando un conjunto de vocabulario que se utilizará para entrenar el modelo LSTM. Este paso es crucial ya que define el espacio de palabras que el modelo podrá utilizar para generar nuevas descripciones de imágenes.

Posteriormente, las descripciones se transforman en una lista de secuencias de texto mediante la función *to_list()*, que reorganiza el diccionario de descripciones en una lista de todas las descripciones disponibles. Este formato es más adecuado para la tokenización, un proceso en el que el texto se convierte en una secuencia de números, ya que los modelos de aprendizaje profundo no pueden procesar texto directamente. Para realizar la tokenización, se utiliza la función *create_tokenizer()*, que toma la lista de descripciones y utiliza el *Tokenizer* de Keras para asignar un índice numérico a cada palabra en el vocabulario.

Finalmente, es necesario ajustar la longitud de las secuencias para asegurar que todas tengan la misma longitud. Esto se realiza utilizando la función *pad_sequences()* de Keras. El *padding* es el proceso de agregar ceros al final de las secuencias para que todas tengan la misma longitud, ya que los modelos LSTM requieren entradas de secuencias de longitud fija.

LSTM (Long Short-Term Memory networks)

Para alimentar eficazmente al modelo LSTM con datos, se requiere un generador de datos que gestione la carga y preparación de los datos de manera eficiente. *CustomDataGenerator* es una clase diseñada para este propósito, creando lotes de datos que aseguran que tanto las características visuales como las secuencias de texto se entreguen en el formato adecuado para el modelo LSTM.

Durante la inicialización, el generador recibe un *DataFrame* que contiene los identificadores de las imágenes y las descripciones asociadas, junto con varios parámetros como el tamaño del lote, el *tokenizer* para el procesamiento de texto, el tamaño del vocabulario (*vocab_size*), la longitud máxima de las secuencias (*max_length*), y las características extraídas de las imágenes (*features*).

Dentro de la clase encontramos cuatro métodos, el método `*on_epoch_end*` baraja los datos si *shuffle* es verdadero, para evitar patrones basados en el orden. El método `*__len__*` calcula el número total de lotes por época, `*__getitem__*` extrae un lote de datos y llama a `*__get_data*` para procesarlos, obteniendo características de imágenes y descripciones, convirtiendo estas últimas en secuencias numéricas con el *tokenizer*, y preparando datos con *padding* y codificación *one-hot*. Finalmente, el método devuelve las características de las imágenes y las secuencias de texto procesadas en el formato adecuado para el entrenamiento del modelo.

Una vez generados los datos por el generador el proceso comienza con la entrada de las características visuales, obtenidas de la CNN VGG16 y se ajustan para que tengan una dimensión adecuada. Estas características se pasan a través de una capa densa con 256 unidades y una función de activación *ReLu*, lo que reduce la dimensionalidad y permite una representación más manejable. Posteriormente, se redimensionan estas características a una forma de (1, 256), preparándolas para la integración con la entrada del modelo LSTM.

El modelo también acepta secuencias de texto, representadas mediante índices numéricos, como entrada adicional. Estas secuencias se convierten en vectores densos mediante una capa de *embedding* con una dimensión de 256, lo que permite transformar los índices de las palabras en representaciones continuas y densas que el modelo puede procesar. Cada palabra del vocabulario tiene un vector de *embedding* asociado y a medida que el modelo se entrena para predecir la siguiente palabra en una secuencia, los vectores de los *embeddings* se ajustan para capturar mejor las relaciones contextuales entre las palabras.

Una vez que se han preparado las características visuales y textuales, estas se concatenan en una única secuencia. Esta combinación de características visuales y textuales se alimenta a una capa LSTM, que se encarga de capturar las dependencias temporales en el texto y de integrar la información visual y textual de manera coherente. Para prevenir el sobreajuste y mejorar la generalización del modelo, se aplica una capa de *Dropout* con una tasa del 50% a la salida del LSTM. Esta salida se suma con las características visuales originales para mantener la información visual en la representación final. Posteriormente, se pasa a través de una capa densa con 128 unidades y activación *ReLu*, seguida de otra capa de *Dropout* para una mayor regularización.

Finalmente, la salida del modelo es una capa densa con una función de activación *softmax* que produce las probabilidades para cada palabra en el vocabulario, permitiendo al modelo generar la descripción textual final de la imagen. La arquitectura se compila utilizando la función de pérdida *categorical_crossentropy* y el optimizador *Adam* [78].

La función de pérdida *categorical_crossentropy* se utiliza para medir la discrepancia entre las predicciones del modelo y las etiquetas verdaderas. Esta función cuantifica el error en las predicciones, guiando el ajuste de los parámetros del modelo durante el entrenamiento para mejorar la precisión en la generación de descripciones.

El optimizador Adam se utiliza para el descenso del gradiente. Se encarga de actualizar los pesos del modelo durante el entrenamiento. Combina las ventajas de otros optimizadores al adaptar la tasa de aprendizaje para cada parámetro y ajustar las actualizaciones basadas en el momento, lo que ayuda a acelerar el entrenamiento y alcanzar un rendimiento óptimo.

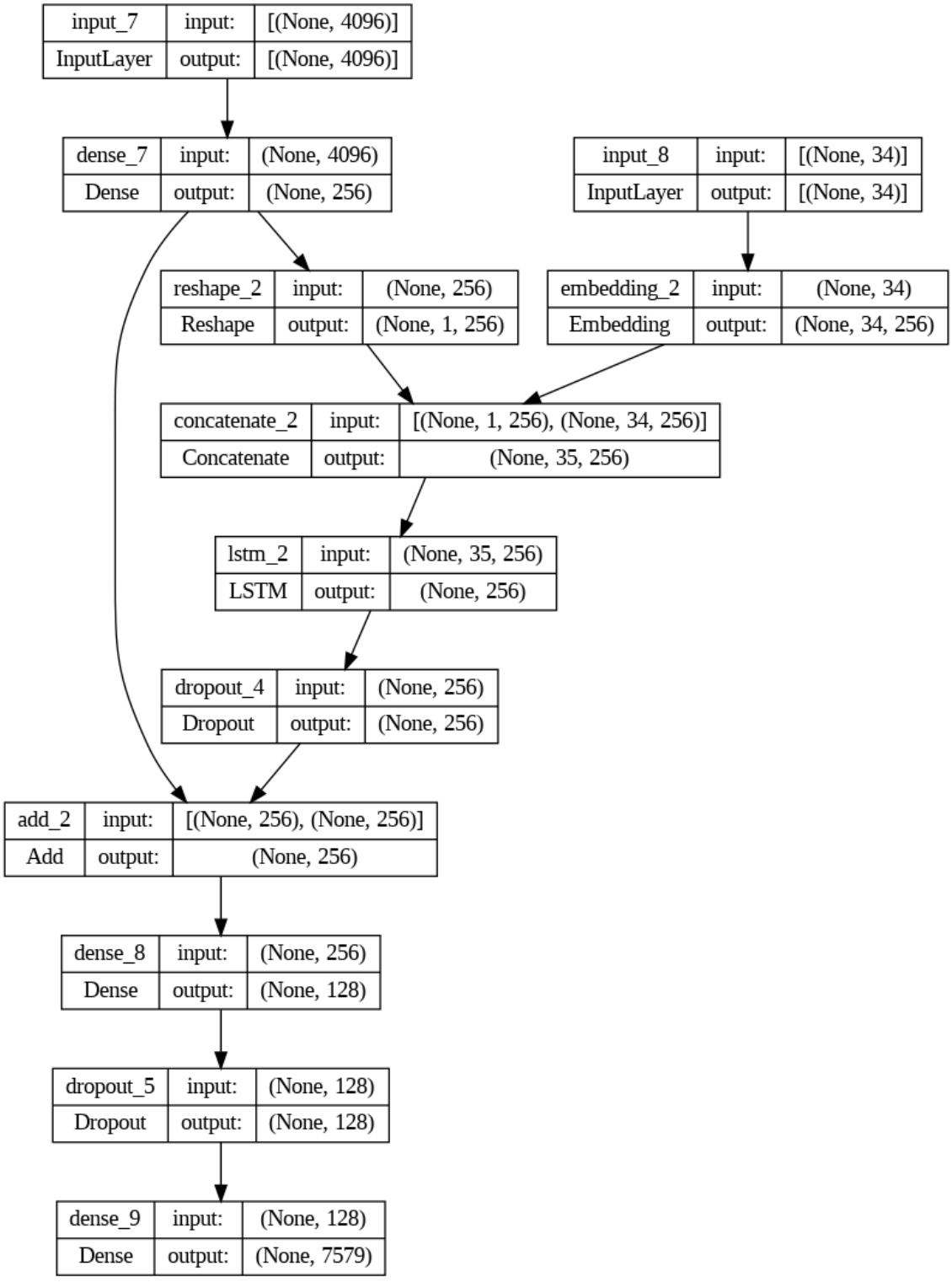

Ilustración 15: Diagrama de flujo LSTM

Entrenamiento

En la implementación del modelo se han integrado varias técnicas para optimizar el entrenamiento y prevenir el sobreajuste. Se utiliza *ModelCheckpoint* para guardar el modelo con la menor pérdida de validación. *EarlyStopping* detiene el entrenamiento si la pérdida de validación no mejora durante cinco épocas consecutivas y restaura los pesos del mejor momento. Finalmente, *ReduceLROnPlateau* ajusta la tasa de aprendizaje si la pérdida de validación no mejora durante tres épocas, reduciendo la tasa de aprendizaje para optimizar el proceso sin disminuirla demasiado. Estas estrategias colaboran para mejorar el rendimiento y evitar el sobreajuste del modelo.

El modelo se entrena con una duración de 50 épocas. Durante el proceso, el rendimiento del modelo se evalúa en el conjunto de validación.

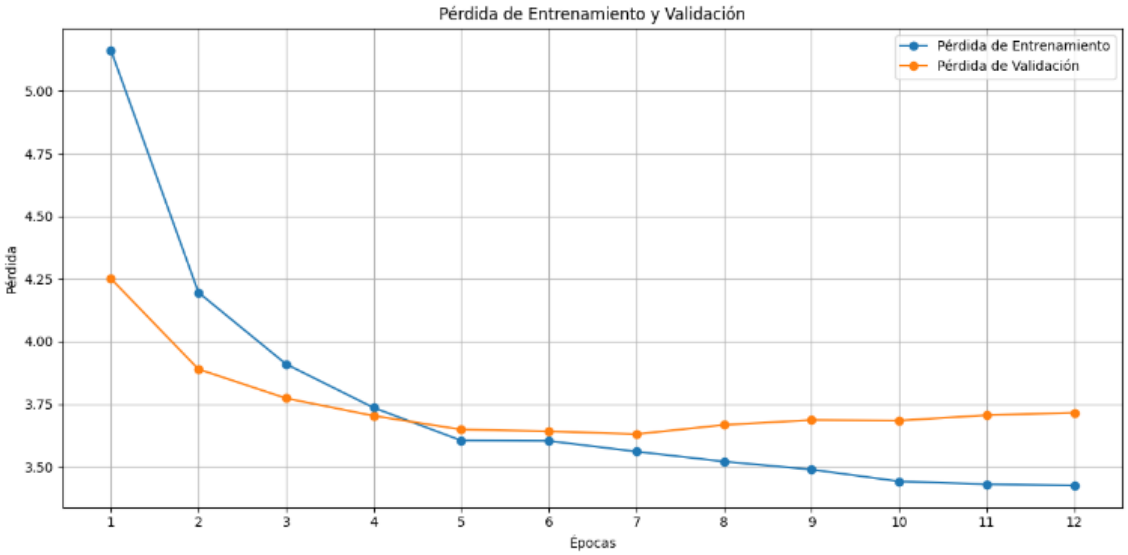

Ilustración 16: Monitorización del entrenamiento de la Arquitectura 1

Durante las 12 épocas de entrenamiento, la pérdida de entrenamiento (*train_loss*) mostró una reducción constante, pasando de 5.1630a 3.425. Sin embargo, la pérdida de validación (*val_loss*) disminuyó hasta el séptimo período (3.630) antes de comenzar a aumentar nuevamente, alcanzando 3.7156 en la última época. Este comportamiento sugiere que el modelo empezó a sobre ajustarse a los datos de entrenamiento después de la séptima época. En la época 10 *ReduceLROnPlatau* redujo la tasa de aprendizaje debido a que no se observó mejora en la pérdida de validación durante tres épocas consecutivas. Posteriormente, en la época 12, se aplicó *EarlyStopping* para detener el entrenamiento al no haber mejorado la pérdida de validación.

Finalmente, el modelo entrenado alcanzó un valor de pérdida de 3.425 y una pérdida de validación de 3.63 en la séptima época.

6.1.4 Resultados

Para evaluar el rendimiento del modelo, se ha optado por utilizar la métrica BLEU, que mide la calidad de las descripciones generadas comparadas con las descripciones de referencia. Los resultados obtenidos son los siguientes:

	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Modelo CNN - LSTM	61.8	38.2	28.1	15.1

Tabla 3: Resultados BLEU en la Arquitectura 1

Estos valores indican el grado de coincidencia entre las descripciones generadas y las referencia. Los resultados obtenidos con la métrica BLEU muestran un desempeño variado en la generación de descripciones para las imágenes. El valor de BLEU-1, que es 61.8, sugiere que las descripciones generadas tienen una coincidencia relativamente alta con las descripciones de referencia a nivel de unigramas. Esto indica que el modelo es capaz de capturar y replicar palabras individuales con una buena precisión.

Sin embargo, aun siendo normal que los valores desciendan BLEU-2, BLEU-3 y BLEU-4, que son 38.2, 28.1 y 15.1 respectivamente, revelan una disminución considerable en la coincidencia a medida que se consideran secuencias de palabras más largas. Esto sugiere que, aunque el modelo puede generar palabras correctas, tiene dificultades para mantener la coherencia y la precisión en secuencias de mayor longitud.

El descenso en los valores de BLEU para n-gramas más largos podría indicar que el modelo enfrenta desafíos en la construcción de descripciones que sean tanto contextualmente precisas como coherentes a lo largo de frases completas. Este comportamiento es común en modelos de generación de texto que pueden aprender patrones a nivel de palabras individuales, pero tienen más dificultades con la coherencia global y la estructura gramatical de las frases

Al observar las predicciones del modelo, se puede notar que las imágenes que contienen perros son descritas con notable precisión, identificando también correctamente contextos como agua, hierba o nieve. No obstante, este éxito viene acompañado de problemas significativos. El modelo tiende a sobre generalizar, clasificando incorrectamente a otros animales como perros o cualquier superficie verde como hierba.

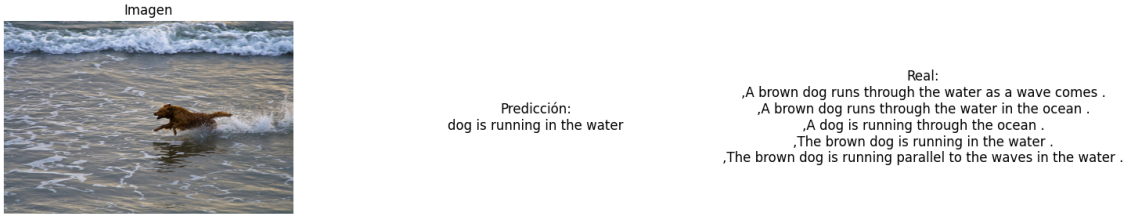

Ilustración 17: Descripciones referencia y generada por la Arquitectura 1

Este comportamiento se debe, en gran parte, a la naturaleza del conjunto de datos de entrenamiento, que incluye una gran cantidad de imágenes de perros y paisajes con hierba, nieve o agua. Como resultado, el modelo ha aprendido a asociar estas características comunes con términos descriptivos específicos, lo que lleva a una sobrerrepresentación de estos elementos en las descripciones generadas. La tendencia a generalizar excesivamente refleja las limitaciones del modelo para distinguir entre características similares, resultado de una representación insuficiente y la falta de diversidad en los datos.

Se observa también que el modelo tiene dificultades para captar detalles precisos en las imágenes, lo que resulta en descripciones que son demasiado generales y carecen de especificidad. Por ejemplo, no identifica correctamente la cantidad de objetos en una escena, a menudo mencionando uno o dos cuando hay más. También comete errores al describir colores, asignando, por ejemplo, el color rojo a objetos que en realidad son de otro color.

Este comportamiento indica un claro sobreajuste a ciertas características del conjunto de datos de entrenamiento. El modelo, en lugar de adaptarse bien a nuevos ejemplos, se apoya demasiado en patrones frecuentes que ha aprendido durante el entrenamiento. Esto provoca descripciones repetitivas y erróneas, especialmente cuando se enfrenta a imágenes que presentan variaciones no tan comunes en los datos originales, como animales diferentes o colores menos predominantes.

Por ejemplo, debido a que el conjunto de datos de entrenamiento incluye una gran cantidad de imágenes de perros, el modelo ha aprendido a asociar la presencia de animales con este concepto, lo que provoca que describa casi todos los animales como "perros", aunque no lo sean. De manera similar, el modelo ha aprendido a asociar ciertos colores, como el rojo, con objetos en general, lo que lo lleva a describir objetos de diferentes colores como "rojo", incluso si no lo son en realidad.

Para abordar estas limitaciones, el uso de modelos basados en *Transformers*, se ha convertido en una estrategia eficaz. Los *Transformers* son especialmente buenos para captar detalles finos en las imágenes debido a su capacidad de atención distribuida, que permite al modelo enfocarse en múltiples aspectos de la imagen simultáneamente. Esta atención detallada facilita la generación de descripciones más precisas y específicas, mejorando no solo la identificación de la cantidad de objetos, sino también la detección de colores y otros atributos visuales sutiles. Al utilizar mecanismos de atención, los modelos basados en *Transformers* pueden reconocer mejor los elementos clave y sus interacciones en la imagen, proporcionando descripciones más ricas y detalladas que reflejan con mayor exactitud el contenido visual.

A continuación, en las siguientes páginas se muestran ejemplos de descripciones tanto correctas como incorrectas generadas por este primer modelo:

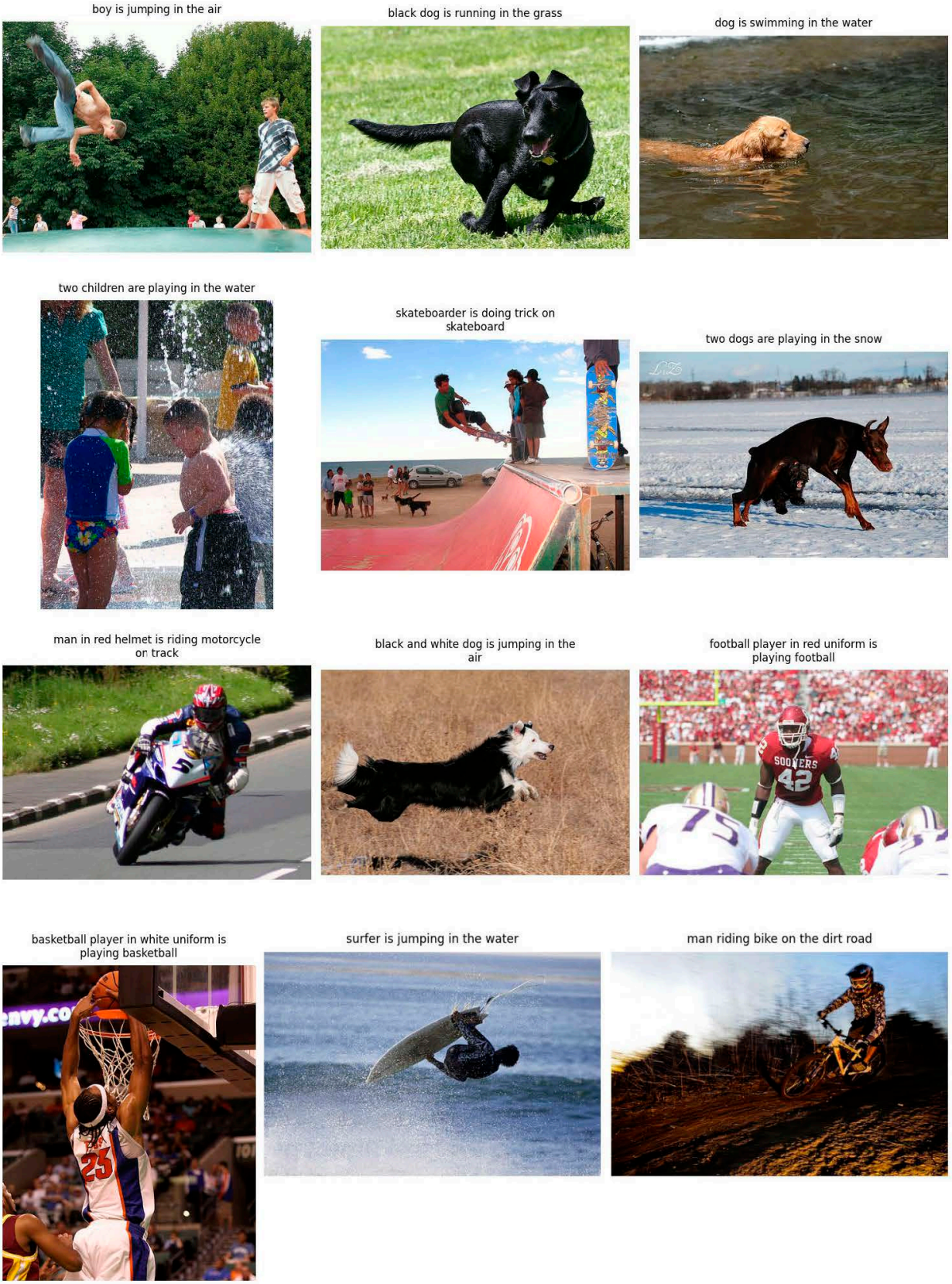

Ilustración 18: Imágenes bien descritas por la Arquitectura 1

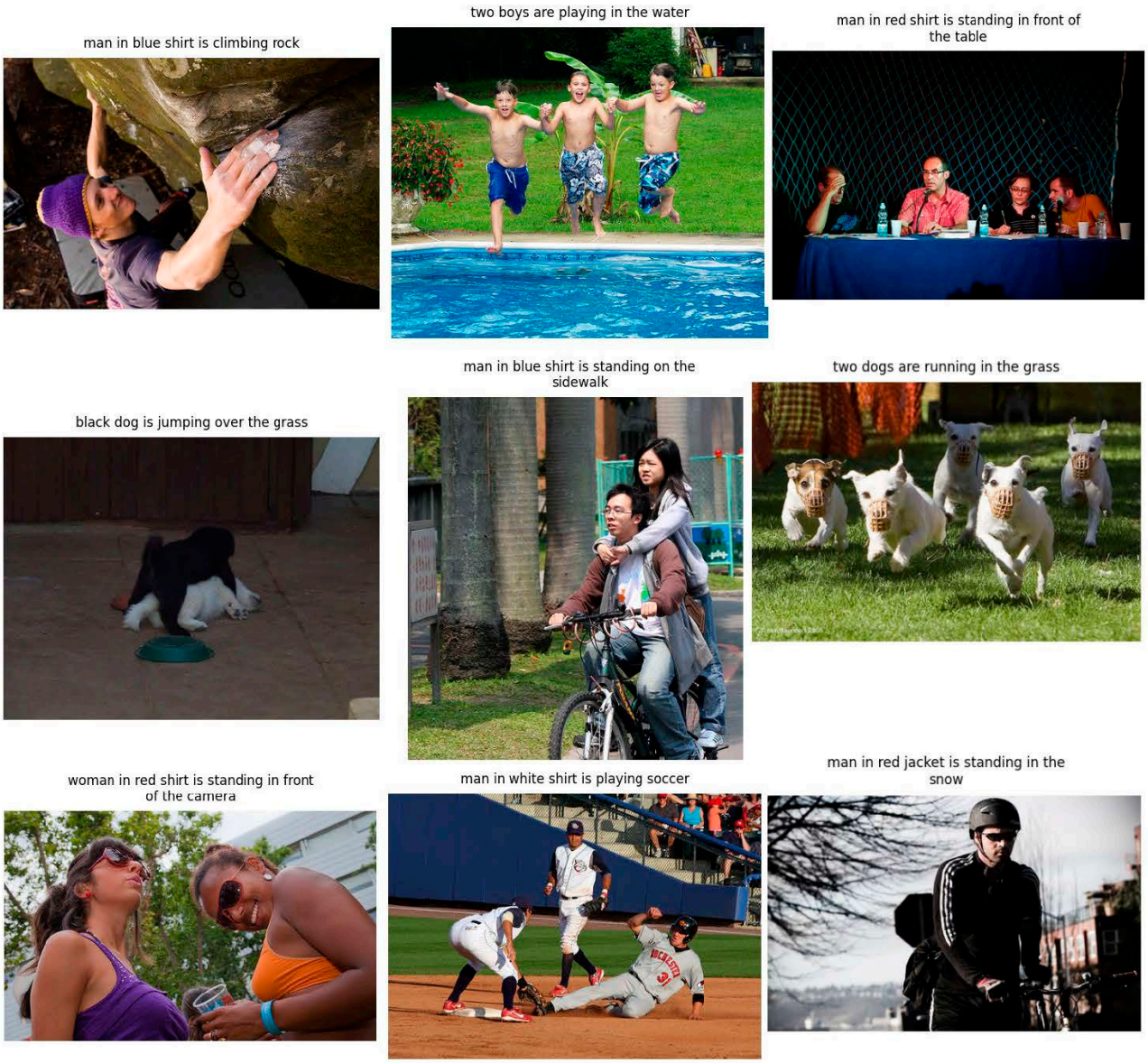

Ilustración 19: Imágenes mal descritas por la Arquitectura 1

En estas imágenes se pueden apreciar claramente los errores mencionados anteriormente. Por ejemplo, el modelo identifica incorrectamente a "dos niños jugando en el agua" cuando en realidad hay tres, o señala "dos perros" cuando en la escena aparecen cinco. Además, se observan errores relacionados con la identificación de colores: en la primera imagen, el modelo describe la camiseta como azul, y en la última imagen, clasifica erróneamente una chaqueta negra como roja. También hay fallos en la detección del entorno, como cuando el modelo describe la presencia de hierba o nieve en escenas donde no existen dichos elementos.

6.2 Arquitectura 2: *DualTransformer* (ViT + GPT-2)

La segunda arquitectura implementada combina el *Vision Transformer* (ViT) y GPT-2. El *Vision Transformer* extrae las características de las imágenes, generando un vector de características que representa la información visual global. Este vector se pasa a GPT-2, que utiliza esta representación para producir descripciones textuales detalladas y coherentes. Esta combinación permite aprovechar la capacidad del ViT para captar detalles visuales complejos y la habilidad de GPT-2 para generar descripciones en lenguaje natural.

6.2.1 Bibliotecas

En esta segunda arquitectura se utilizan las bibliotecas *NumPy, Pickle, Pandas, OS, Tqdm, string, matplotlib, random* y *NLTK*, que ya han sido utilizadas en la arquitectura anterior. Otras bibliotecas utilizadas son:

- **Load_metric** de la biblioteca *datasets* se utiliza para cargar la métrica BLEU.
- **PIL.Image** para la manipulación y procesamiento de imágenes.
- **Multiprocessing** permite la ejecución de tareas en paralelo.
- **Sklearn.model_selection** para dividir los datos en conjuntos de entrenamiento y prueba.
- **Zipfile** y **shutil** para guardar y cargar el modelo.

Para la creación del modelo hemos hecho uso de las librerías *Torch* y *Transformers. Torch* nos proporciona un marco flexible y eficiente para la construcción y entrenamiento de redes neuronales, permitiendo un manejo intuitivo de tensores y operaciones en GPU. Por otro lado, la librería *Transformers* nos ofrece acceso a una amplia gama de modelos preentrenados de procesamiento del lenguaje natural y visión por computador, facilitando la integración de un codificador de visión (ViT) y un decodificador de lenguaje (GPT-2) en nuestro proyecto.

6.2.2 Preprocesado de datos

El preprocesado de texto en esta segunda arquitectura difiere del de la primera al utilizar el tokenizador GPT-2, que convierte las descripciones de texto en secuencias de identificadores numéricos. En primer lugar, se comienza cargando el archivo *Flickr8K*.token.txt, que contiene las descripciones asociadas a cada imagen y se utilizan las funciones *load_captions()* y *clean_captions()* ya usadas en la arquitectura anterior. Esto proporciona descripciones de texto limpias y normalizadas que se utilizarán para el entrenamiento del modelo. Estas descripciones, junto con sus imágenes, se convierten en un *Dataframe* de pandas y se dividen en conjuntos de entrenamiento, validación y prueba.

Posteriormente, se utiliza la clase *ImgDataset* para preparar los datos para el modelo. En esta clase, el método *__getitem__* convierte cada imagen y su descripción en tensores adecuados para el modelo, utilizando el *feature_extractor* para preparar y transformar las imágenes para que puedan ser procesadas correctamente por el modelo *Vision Transformer* (ViT). Simultáneamente, el *tokenizer* se utiliza para procesar el texto. Las descripciones se tokenizan, se rellenan mediante *padding* para alcanzar la longitud máxima y se convierten en IDs de tokens, preparando así los datos para el entrenamiento del modelo.

En esta arquitectura, se emplean dos componentes diferentes para el preprocesamiento de los datos frente a la anterior arquitectura.

- **Feature Extractor**: Se utiliza ViTFeatureExtractor para preparar las imágenes de entrada al modelo *Vision Transformer* (ViT). Este extractor ajusta el tamaño de las imágenes, las normaliza según los requisitos del modelo preentrenado, y las convierte en tensores PyTorch adecuados para ser procesados por el modelo de visión.
- **Tokenizador**: El tokenizador de GPT-2 convierte las descripciones de texto en secuencias numéricas (tokens) que el modelo puede procesar. Este tokenizador descompone el texto en tokens individuales y aplica padding para uniformar la longitud de las secuencias.

6.2.3 Parámetros y configuración del modelo

Antes de comenzar el entrenamiento del modelo, se deben configurar los parámetros y ajustes necesarios para el proceso.

Configuración de parámetros:

- Codificador (CODIFICADOR): google/vit-base-patch16-224
- Decodificador (DECODIFICADOR): gpt2
- Tamaño del lote de entrenamiento (TRAIN_BATCH_SIZE): 8
- Tamaño del lote de validación (VAL_BATCH_SIZE): 16
- Número de épocas de validación (VAL_EPOCHS): 1
- Tasa de aprendizaje (LR): 3e-5
- Semilla aleatoria (SEED): 42
- Longitud máxima de secuencia (MAX_LEN): 34
- Longitud del resumen (SUMMARY_LEN): 25
- Decaimiento del peso (WEIGHT_DECAY): 0.01
- Media de normalización (MEAN): (0.485, 0.456, 0.406)
- Desviación estándar de normalización (STD): (0.229, 0.224, 0.225)
- Porcentaje de datos de entrenamiento (TRAIN_PCT): 0.95
- Número total de épocas (EPOCHS): 3
- Tamaño de imagen (IMG_SIZE): (224, 224)
- Etiqueta de máscara (LABEL_MASK): -100
- TOP_K: 1000
- TOP_P: 0.95

Configuración del modelo (model.config):

- Token de inicio del decodificador (start_token_id): *tokenizer*.bos_token_id
- Token de relleno (pad_token_id): *tokenizer*.pad_token_id
- Tamaño del vocabulario (vocab_size): model.config.decodificador.vocab_size
- Token de fin de secuencia (eos_token_id): *tokenizer*.sep_token_id
- Longitud máxima de la secuencia (max_length): 34
- EarlyStopping: True
- Tamaño de n-gramas sin repetición (no_repeat_ngram_size): 3
- Penalización por longitud (length_penalty): 2.0
- num_beams: 3

Argumentos de entrenamiento:

- Generar durante la predicción (predict_with_generate): True
- Estrategia de evaluación (evaluation_strategy): "epoch"
- Realizar entrenamiento (do_train): True
- Realizar evaluación (do_eval): True
- Pasos de registro (logging_steps): 1024
- Pasos de guardado (save_steps): 2048
- Pasos de calentamiento (warmup_steps): 1024
- Sobrescribir el directorio de salida (overwrite_output_dir): True

El Codificador (CODIFICADOR) utilizado es *google/vit-base-patch16-22*[7], mientras que el Decodificador (DECODIFICADOR) es *gpt*[8]. El tamaño del lote de entrenamiento se establece en 8, mientras que el de validación en 16, permitiendo un equilibrio entre eficiencia y uso de memoria. El tamaño de lote se refiere al número de muestras que el modelo procesa antes de actualizar sus pesos. Se define un número total de 3 épocas para el entrenamiento, con una tasa de aprendizaje de 3e-5 para ajustar los pesos del modelo. La longitud máxima de la secuencia se limita a 34 tokens y se aplica una penalización por longitud de 2 para evitar descripciones excesivamente largas. Además, se emplea *beam search* con un tamaño de *beam* 3, lo que significa que el modelo mantiene y evalúa las tres secuencias más prometedoras en cada paso de generación. Esto permite explorar múltiples opciones y seleccionar las secuencias con mayor probabilidad.

La parada anticipada está activada para prevenir el sobreajuste, deteniendo el entrenamiento si no se observan mejoras en la validación. Además, se emplea una tasa de decaimiento del peso de 0.01 para regularizar el modelo y evitar que se ajuste demasiado a los datos de entrenamiento. La normalización de las imágenes se realiza con una media de (0.485, 0.456, 0.406) y una desviación estándar de (0.229, 0.224, 0.225), que son valores predefinidos para el modelo *Vision Transformer* (ViT). El modelo está configurado para guardar el mejor estado durante el entrenamiento, asegurándose que solo se mantenga la versión más reciente del modelo. La estrategia de evaluación se establece para que ocurra al final de cada época.

[7] https://huggingface.co/google/vit-base-patch16-224
[8] https://huggingface.co/openai-community/gpt2

6.2.4 Entrenamiento

Una vez que los datos han sido preparados, y los parámetros configurados, se procede a crear el modelo, integrando un codificador de visión (ViT) y un decodificador de lenguaje (GPT-2).

Para iniciar el entrenamiento, se configura el optimizador *Adam* y se establece un *scheduler* de tasa de aprendizaje lineal que ajusta la tasa de aprendizaje a lo largo de las épocas. El *scheduler* se configura para realizar un calentamiento inicial durante un número definido de pasos, seguido de una disminución lineal de la tasa de aprendizaje.

El *trainer* se instancia con el modelo, el optimizador, el *scheduler* y otros parámetros de entrenamiento. Se especifica que la métrica utilizada para la evaluación durante el entrenamiento es BLEU, a través de la función *compute_metrics*.

Dado que el proceso de entrenamiento para esta arquitectura es significativamente más costoso que para la anterior, se decidió limitar el entrenamiento a solo 3 épocas. Inicialmente, se establecieron el tamaño del lote de validación en 8 y la tasa de aprendizaje en 5e-5. Durante el entrenamiento, se observó un aumento en la pérdida de validación a partir de la segunda época, lo que sugería un posible sobreajuste del modelo a los datos de entrenamiento.

Para abordar este problema, se ajustaron hiperparámetros: se aumentó el tamaño del lote de validación a 16 y se redujo la tasa de aprendizaje a 3e-5. Al incrementar el tamaño del lote, se obtienen estimaciones de gradientes más estables. Esto se debe a que un lote más grande proporciona una media más representativa del gradiente sobre el conjunto de datos. Al disminuir la tasa de aprendizaje de 5e-5 a 3e-5, el modelo realiza ajustes más pequeños en los pesos durante cada iteración. Estos cambios se implementaron con el objetivo de mejorar la capacidad del modelo para generalizar a datos no vistos y reducir la pérdida de validación.

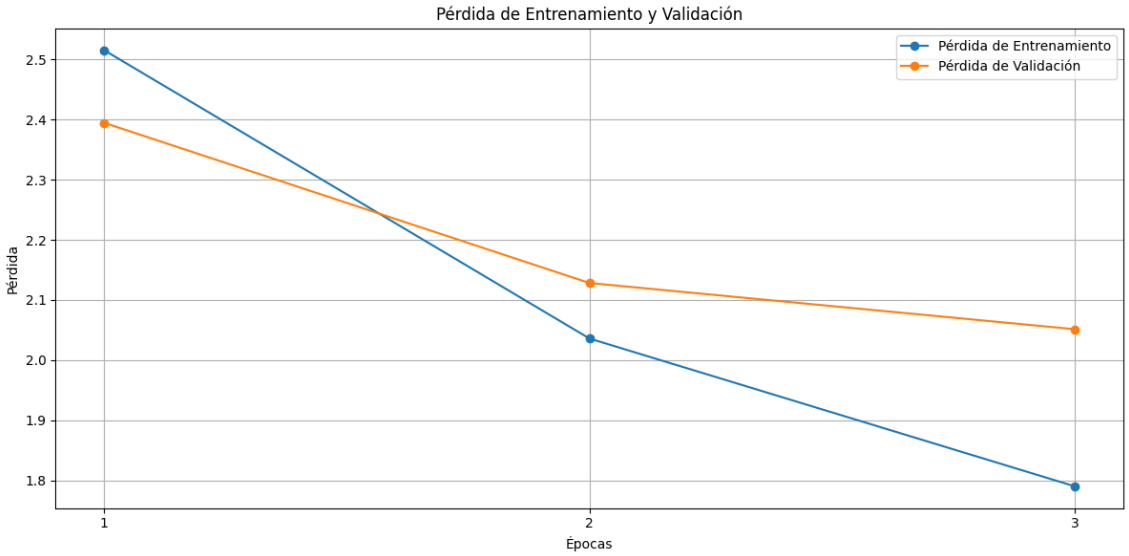

Ilustración 20: Monitorización del entrenamiento de la Arquitectura 2

A lo largo de las 3 épocas, se observó una disminución continua en la pérdida de entrenamiento (train_loss), que descendió de 2.5156 en la primera época a 1.79 en la tercera. La pérdida de validación (val_loss) también mostró una reducción constante, bajando de 2.39478 en la primera época a 2.0514 en la tercera. Estos resultados indican que el modelo no solo está aprendiendo de manera efectiva durante el entrenamiento, sino que también está generalizando bien a los datos de validación.

Sin embargo, la tasa de mejora en la pérdida de validación se desacelera en las últimas épocas. La disminución es más pronunciada entre la primera y la segunda época en comparación con la segunda y la tercera. Este patrón sugiere que el modelo está alcanzando un punto de optimización donde las mejoras adicionales son más difíciles de lograr. En consecuencia, el modelo parece estar convergiendo, y continuar el entrenamiento podría resultar en mejoras marginales.

6.2.5 Resultados

Los resultados BLEU obtenidos en esta segunda arquitectura son los siguientes:

	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Modelo CNN - LSTM	61.8	38.2	28.1	15.1
Modelo ViT - GPT2	67.7	47.8	37.8	23.6

Tabla 4: Resultados BLEU en Arquitectura 1 y 2

Estos valores indican una mejora significativa respecto al modelo anterior. Los resultados muestran que el modelo tiene un buen desempeño en captar términos individuales y pares de palabras, con una puntuación de 67.7 en BLEU-1 y 47.8 en BLEU-2. Sin embargo, su rendimiento disminuye para secuencias más largas, con puntuaciones de 37.8 en BLEU-3 y 23.6 en BLEU-4.

Observando las descripciones generadas por el nuevo modelo, se evidencia una mejora significativa en comparación con el modelo anterior. Esta mejora se refleja no solo en la capacidad del modelo para detectar más elementos dentro de las imágenes, sino también en la calidad y precisión de las descripciones generadas. El modelo previo, que combinaba una arquitectura CNN+LSTM, mostraba ciertas limitaciones en la interpretación visual detallada, especialmente en la identificación precisa de la cantidad y tipo de objetos presentes. Por ejemplo, si una imagen contenía tres perros, el modelo anterior a menudo fallaba al reconocer el número exacto, proporcionando una descripción más vaga y generalizada.

En contraste, el modelo actual, demuestra una capacidad mejorada para captar detalles específicos y contextuales de las imágenes. Este avance se debe en gran parte a los mecanismos de atención que permiten al modelo enfocarse dinámicamente en diferentes partes de la imagen mientras genera descripciones textuales. Así, si una

imagen contiene un perro de una raza específica como un *golden retriever*, el modelo no solo reconoce la presencia de un perro, sino que a veces incluso también identifica correctamente la raza, proporcionando una descripción mucho más precisa y detallada.

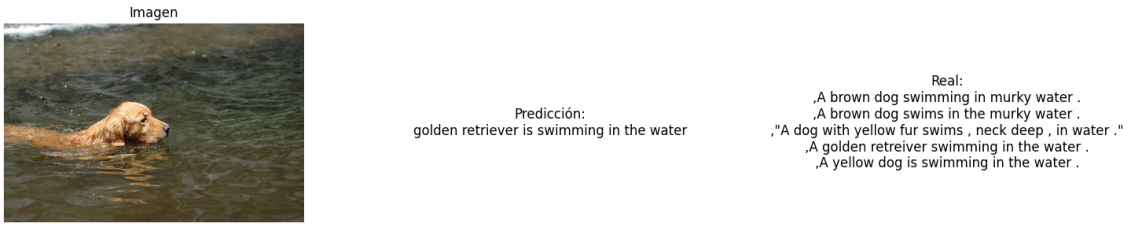

Ilustración 21: Descripciones referencia y generada por la Arquitectura 2

Sin embargo, a pesar de la precisión mejorada en las descripciones generadas por el modelo, se observa que el valor de la métrica BLEU puede ser relativamente bajo en algunos casos. Esto ocurre porque la métrica BLEU se basa en la coincidencia exacta de palabras y frases entre las descripciones generadas y las descripciones de referencia. En situaciones donde el modelo ofrece una descripción precisa, pero utiliza terminología diferente a la de las descripciones de referencia, el valor de BLEU puede ser penalizado.

Este fenómeno pone de relieve una limitación inherente al uso exclusivo de métricas basadas en coincidencias exactas como BLEU para evaluar modelos de generación de texto. Por lo tanto, los resultados sugieren que, a pesar de un BLEU no excesivamente alto, el modelo ha avanzado significativamente en su capacidad para comprender y describir imágenes de manera más precisa y detallada, capturando mejor la esencia de lo que se muestra.

A continuación, en las siguientes páginas se muestran ejemplos de imágenes generadas por este segundo modelo:

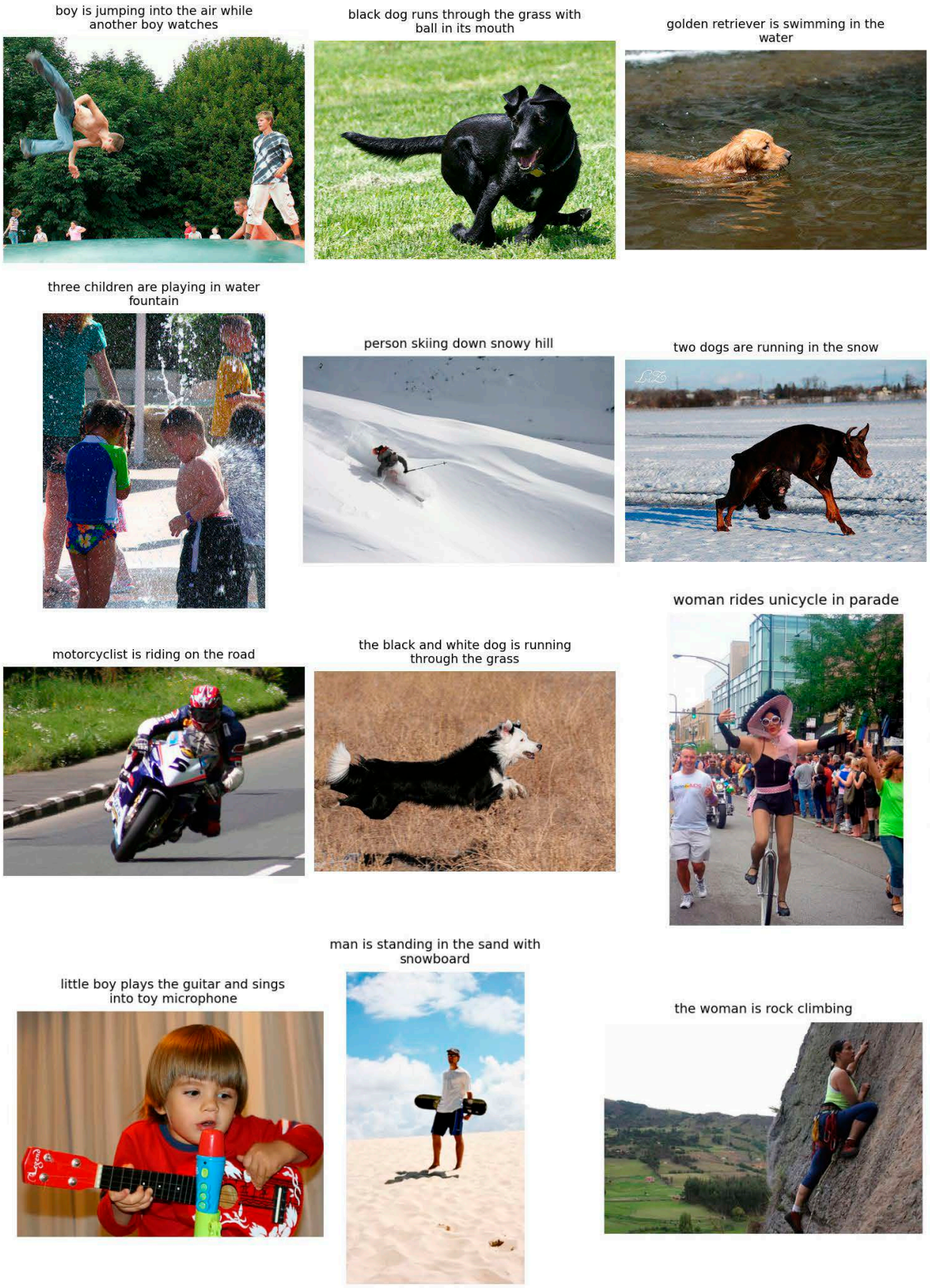

Ilustración 22: Ejemplo 1: Imágenes bien descritas por la Arquitectura 2

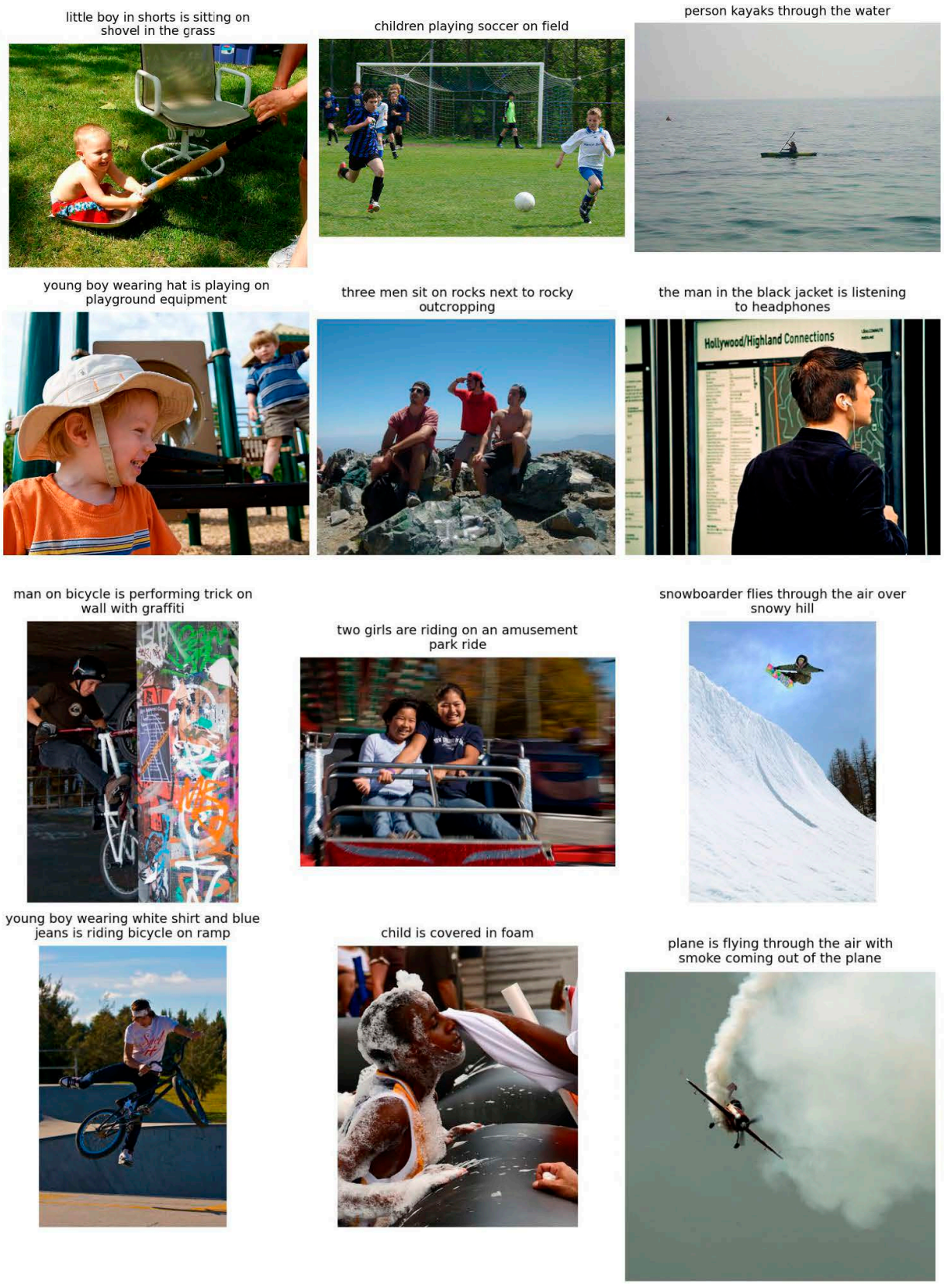

Ilustración 23: Ejemplo 2: Imágenes bien descritas por la Arquitectura 2

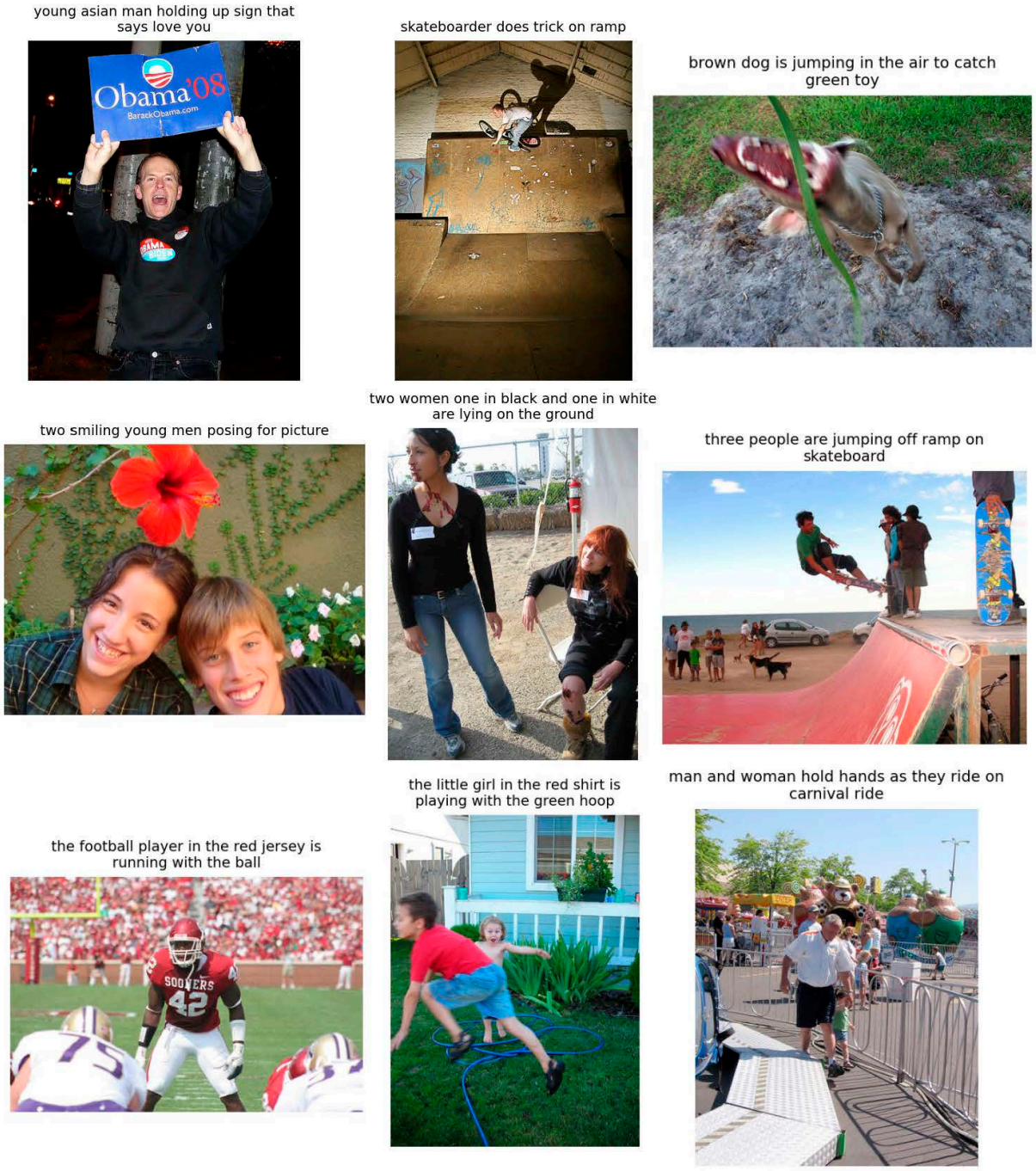

Ilustración 24: Imágenes mal descritas por la Arquitectura 2

En esta arquitectura, se observa un nivel de detalle significativamente mejorado en comparación con la arquitectura anterior. Sin embargo, el modelo aún presenta algunos errores notables en la generación de descripciones. Por ejemplo, en la segunda imagen, el modelo describe incorrectamente a una persona montando en skate, cuando en realidad la persona está en una bicicleta. En otra instancia, en la cuarta imagen, el modelo identifica a dos hombres jóvenes, cuando en realidad hay un hombre y una mujer. Estos errores, aunque relativamente menores, destacan la necesidad de perfeccionar el modelo. Aunque las descripciones generadas son mayoritariamente precisas y detalladas, estas inconsistencias indican que aún hay margen para mejorar. Es necesario ajustar y refinar el modelo para aumentar su capacidad de interpretar con

mayor precisión el contenido visual y reducir la frecuencia de errores. La mejora continua en estos aspectos permitirá que el modelo proporcione descripciones más coherentes y precisas, acercándose cada vez más a una interpretación fiel del contenido de las imágenes.

7. Conclusiones

La comparación entre los dos modelos utilizados para la generación de descripciones de imágenes, revela diferencias significativas en cuanto a su capacidad para interpretar y describir imágenes con precisión y detalle.

El modelo CNN+LSTM, que utiliza redes neuronales convolucionales para la extracción de características visuales y redes neuronales recurrentes para la generación de texto, mostró un desempeño decente en la creación de descripciones. Sin embargo, este enfoque tradicional tiene limitaciones en la captación de detalles finos y en la generación de descripciones más elaboradas, ya que las RNN pueden tener dificultades para gestionar dependencias a largo plazo y para integrar eficazmente la información contextual de toda la imagen.

Por otro lado, el modelo basado en *Transformers*, que combina ViT para la comprensión de imágenes y GPT-2 para la generación de texto, demostró ser más efectivo y preciso. Gracias a su capacidad para procesar simultáneamente todas las partes de una imagen y considerar todas las palabras previamente generadas al crear una descripción, este modelo captó detalles más específicos y generó descripciones más ricas y coherentes. Además, el uso de mecanismos de atención permitió que el modelo se enfocara en las características más relevantes de la imagen, mejorando la precisión en la identificación de objetos y detalles específicos, como la raza de un perro en lugar de una descripción general.

En términos de rendimiento medido por las métricas BLEU, el modelo basado en *Transformers* logró un mejor alineamiento con las descripciones de referencia, aunque con ciertas penalizaciones en los casos donde el modelo era más detallado que las descripciones de referencia. A pesar de esto, la superioridad del modelo *Transformer* en la captación de detalles y la generación de texto más fluido y específico sugiere un avance significativo respecto al enfoque tradicional de CNN+LSTM. Esto indica que los modelos basados en *Transformers* son más adecuados para tareas que requieren una comprensión profunda de las imágenes y una capacidad avanzada de generación de lenguaje natural.

Modelo	**BLEU-1**	**BLEU-2**	**BLEU-3**	**BLEU-4**
m-RNN (Mao et al., 2014) [46]	58	28	23	—
m-RNN (Mao et al., 2015b) [47]	56.5	38.6	25.6	17
m-RNN (Karpathy & Fei-Fei, 2015) [48]	57.9	38,3	24.5	16
Modelo CNN - LSTM	**61.8**	**38.2**	**28.1**	**15.1**

Google NIC (Vinyals et al., 2014) [22]	63	41	27	—
Log Bilinear (Kiros et al., 2014) [49]	65.6	42.4	27.7	17.7
Emb-gLSTM (Jia et al., 2015) [50]	64.7	45.9	31.8	21.6
Soft-Attention (Xu et al., 2015) [23]	67	44.8	29.9	19,5
Hard-Attention (Xu et al., 2015) [23]	67	45.7	31.4	21.3
Modelo ViT - GPT-2	**67.7**	**47.8**	**37.8**	**23.6**
Wu et al. 2018 (Wu et al., 2018) [51]	74	54	38	27
ETransCap (Mundu et al., 2024) [52]	76,1	61,3	45,9	40

Tabla 5: Comparación de resultados entre arquitecturas propuestas y trabajos estado del arte en Flickr8K

En la comparación de los resultados BLEU entre ambos modelos con otras arquitecturas conocidas, se observan tendencias notables que destacan el avance y la efectividad del modelo basado en *Transformers*.

El modelo CNN+LSTM, se sitúa en una posición intermedia en la tabla comparativa. Aunque ofrece una mejora considerable respecto a algunos enfoques anteriores, como el m-RNN (Mao et al., 2014), sus resultados están por debajo de modelos más avanzados, como los modelos con mecanismos de atención como era de esperar.

El modelo ViT+GPT-2, no solo supera al modelo CNN+LSTM en todas las métricas BLEU, sino que también iguala o supera el rendimiento de otros enfoques avanzados, como los modelos atencionales. Esta superioridad refleja su capacidad para capturar detalles más finos y generar descripciones más precisas y coherentes.

La comparación sugiere que los modelos basados en *Transformers*, representan un avance significativo sobre los enfoques tradicionales y algunos métodos de atención más antiguos.

Sin embargo, es importante considerar el contexto en el que se han entrenado estos modelos. Ambos han sido entrenados utilizando el conjunto de datos Flickr8K, que, aunque útil, es relativamente pequeño en comparación con otros conjuntos de datos más extensos. Con poco más de 8000 imágenes, Flickr8K puede limitar la diversidad y cantidad de información que el modelo puede aprender. En consecuencia, mientras que los modelos basados en *Transformers* demuestran un rendimiento superior en este conjunto de datos específico, la escala y la diversidad de los datos de entrenamiento pueden influir en la capacidad del modelo para generalizar a nuevas imágenes y situaciones. Esta limitación subraya la importancia de considerar el tamaño y la calidad del conjunto de datos en la evaluación del rendimiento del modelo y su aplicabilidad en escenarios del mundo real.

Al aplicar estos modelos a conjuntos de datos mucho mayores, como aquellos con millones de imágenes, se esperaría una mejora significativa en el rendimiento. Un conjunto de datos más grande permitiría al modelo aprender de una mayor variedad de imágenes y descripciones, resultando en descripciones más precisas y detalladas. Además, se reduciría el riesgo de sobreajuste y se optimizarían los parámetros del modelo, mejorando la capacidad de generalización y la capacidad para capturar detalles finos. En conjunto, esto llevaría a un rendimiento significativamente mejorado en la generación de descripciones.

7.1 Relación del trabajo desarrollado con los estudios cursados

Los conocimientos obtenidos en las asignaturas del grado en Ciencia de Datos en la *Universitat Politècnica de València* (UPV) han sido fundamentales para la realización de este proyecto.

- **Fundamentos de Programación** y **Programación** me ha ofrecido una base sólida en técnicas de programación, cruciales para desarrollar los scripts y algoritmos empleados.

- **Modelos Descriptivos y Predictivos I** y **II** me han proporcionado una comprensión profunda de los modelos de machine *learning* y *deep learning*.

- **Infraestructura para el procesamiento de datos** me ha enseñado cómo estructurar y presentar trabajos académicos de manera profesional.

- **Lenguaje natural y recuperación de la información** me ha proporcionado conocimientos sobre el procesamiento del lenguaje natural y sus métodos asociados.

- En **Evaluación, Despliegue y Monitorización de Modelos** he aprendido conceptos clave en cuanto a la evaluación y el despliegue de modelos predictivos.

- **Proyecto I, II,** y **III** me ha servido de entrenamiento para realizar el TFG, ya que debíamos de realizar un proyecto de una duración de un cuatrimestre acompañados de una memoria similar a la del TFG.

Las asignaturas cursadas en la UPV han proporcionado una formación sólida y conocimientos clave que han sido fundamentales para el desarrollo de este proyecto.

Por último, durante mi estancia de Erasmus en Ostrava (República Checa), la asignatura **Image Analysis** impartida en la *Technical University of Ostrava* (VSB-TUO)[9] me enseñó a extraer y clasificar características visuales utilizando redes neuronales convolucionales (CNN), así como a aplicar técnicas de segmentación y reconocimiento de patrones.

En cuanto a las competencias transversales de la UPV[10], podemos afirmar que estos aspectos se han seguido desarrollando de manera continua a lo largo de la carrera universitaria. En mi TFG, he desarrollado varias competencias transversales clave:

- Comprensión e integración de conocimientos: se han integrado conocimientos adquiridos durante la formación en el grado de programación, análisis de imágenes y modelos predictivos en el desarrollo de mi proyecto.

- Aplicación y pensamiento práctico: se ha recopilado información de las técnicas utilizadas y se han aplicado técnicas prácticas para resolver problemas específicos en la generación automática de descripciones de imágenes.

- Aprendizaje permanente: Ha sido necesario un aprendizaje continuo a lo largo de todo el trabajo y se han aplicado nuevos conocimientos y tecnologías en el campo del análisis de imágenes y deep learning.

- Instrumentación específica: He utilizado herramientas y tecnologías adecuadas, como redes neuronales y modelos de deep learning, para la resolución del problema.

[9] https://www.vsb.cz/en/
[10] https://www.upv.es/entidades/SG/infoweb/sg/info/U0917752.pdf

8. Trabajos futuros

En trabajos futuros se buscará implementar los modelos generadores de descripciones de imágenes utilizando conjuntos de datos más amplios que Flickr8K para lograr resultados superiores. Esto requerirá un aumento en el poder computacional y la aplicación de una gama más amplia de métricas de evaluación para comparar y analizar los modelos de manera más exhaustiva. Además, la inteligencia artificial es un campo en constante evolución que avanza rápidamente cada año, con nuevas técnicas y mejoras continuas, por lo que habrá que mantenerse actualizado con los avances recientes en el campo para integrar las técnicas emergentes y optimizar el rendimiento de los sistemas de generación automática de descripciones.

Bibliografía

[1] MCCULLOCH, Warren S.; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 1943, vol. 5, p. 115-133.

[2] TURING, Alan M. Computing machinery and intelligence. Springer Netherlands, 2009.

[3] ROSENBLATT, Frank. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 1958, vol. 65, no 6, p. 386.

[4] HUTCHINS, W. John; DOSTERT, Leon; GARVIN, Paul. The georgetown-ibm experiment. Machine translation of languages, 1955, p. 124-135.

[5] SAMUEL, Arthur L. Some studies in machine learning using the game of checkers. IBM Journal of research and development, 1959, vol. 3, no 3, p. 210-229.

[6] HORNIK, Kurt; STINCHCOMBE, Maxwell; WHITE, Halbert. Multilayer feedforward networks are universal approximators. Neural networks, 1989, vol. 2, no 5, p. 359-366.

[7] LECUN, Yann, et al. Handwritten digit recognition with a back-propagation network. Advances in neural information processing systems, 1989, vol. 2.

[8] JELINEK, Fred. Self-organized language modeling for speech recognition. Readings in speech recognition, 1990, p. 450-506.

[9] MAYER-SCHÖNBERGER, Viktor; CUKIER, Kenneth. Big data: A revolution that will transform how we live, work, and think. Houghton Mifflin Harcourt, 2013.

[10] KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. ImageNet classification with deep convolutional neural networks. Communications of the ACM, 2017, vol. 60, no 6, p. 84-90.

[11] SUTSKEVER, I. Sequence to Sequence Learning with Neural Networks. arXiv:1409.3215, 2014.

[12] Ibidem Group, 2024. Traducción automática neuronal: Seq2Seq y *Transformers*. [en línea] Disponible en: https://www.ibidemgroup.com/edu/traduccion-automatica-neuronal-seq2seq-*Transformers*/

[13] DEVLIN, Jacob. Bert: Pre-training of deep bidirectional *Transformers* for language understanding. arXiv:1810.04805, 2018.

[14] RADFORD, Alec, et al. Learning transferable visual models from natural language superVision. En International conference on machine learning. PMLR, 2021. p. 8748-8763.

[15] SCHMIDHUBER, Jürgen. Deep learning in neural networks: An overview. Neural networks, 2015, vol. 61, p. 85-117.

[16] LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. nature, 2015, vol. 521, no 7553, p. 436-444.

[17] REDMON, J. You only look once: Unified, real-time object detection. En Proceedings of the IEEE conference on computer Vision and pattern recognition. 2016.

[18] MIKOLOV, Tomas, et al. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 2013, vol. 26.

[19] PENNINGTON, Jeffrey; SOCHER, Richard; MANNING, Christopher D. Glove: Global vectors for word representation. En Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014. p. 1532-1543.

[20] DOSOVITSKIY, Alexey. An image is worth 16x16 words: *Transformers* for image recognition at scale. arXiv:2010.11929, 2020.

[21] STEFANINI, Matteo, et al. From show to tell: A survey on deep learning-based image captioning. IEEE transactions on pattern analysis and machine intelligence, 2022, vol. 45, no 1, p. 539-559.

[22] VINYALS, Oriol, et al. Show and tell: A neural image caption generator. En Proceedings of the IEEE conference on computer Vision and pattern recognition. arXiv:1411.4555, 2014.

[23] XU, Kelvin, et al. Show, attend and tell: Neural image caption generation with visual attention. arXiv:1502.03044, 2015.

[24] WANG, Yiyu; XU, Jungang; SUN, Yingfei. End-to-end *Transformer* based model for image captioning. arXiv:1802.05751, 2022.

[25] PARMAR, Niki, et al. Image *Transformer*. En International conference on machine learning. PMLR, 2018. p. 4055-4064.

[26] RADFORD, A., JIN, P., KERNER, S., et al. Learning Transferable Visual Models From Natural Language SuperVision. arXiv:2103.00020, 2021.

[27] FANG, Hao, et al. From captions to visual concepts and back. En Proceedings of the IEEE conference on computer Vision and pattern recognition. 2015. p. 1473-1482.

[28] MISHRA, Swapneel, et al. Image Caption Generation using Vision *Transformer* and GPT Architecture. En 2024 2nd International Conference on Advancement in Computation & Computer Technologies (InCACCT). IEEE, 2024. p. 1-6.

[29] 9. LI, L., et al. ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks. arXiv:1908.02265, 2019.

[30] ALBAUM, K., et al. Flamingo: A Visual Language Model for Few-Shot Learning. arXiv:2204.14198, 2022.

[31] LI, Bo, et al. Otterhd: A high-resolution multi-modality model. arXiv:2311.04219, 2023.

[32] TOUVRON, Hugo, et al. Llama: Open and efficient foundation language models. arXiv:2302.13971, 2023.

[33] SAOUABE, Abdelkrim, et al. Evolution of Image Captioning Models: An Overview. En 2023 10th International Conference on Wireless Networks and Mobile Communications (WINCOM). IEEE, 2023. p. 1-5.

[34] LIN, Tsung-Yi, et al. Microsoft COCO: Common objects in context. En Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13. Springer International Publishing, 2014. p. 740-755.

[35] YOUNG, Peter, et al. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. Transactions of the Association for Computational Linguistics, 2014, vol. 2, p. 67-78.

[36] HODOSH, Micah; YOUNG, Peter; HOCKENMAIER, Julia. Framing image description as a ranking task: Data, models and evaluation metrics. Journal of Artificial Intelligence Research, 2013, vol. 47, p. 853-899.

[37] RASHTCHIAN, Cyrus, et al. Collecting image annotations using amazon's mechanical turk. En Proceedings of the NAACL HLT 2010 workshop on creating speech and language data with Amazon's Mechanical Turk. 2010. p. 139-147.

[38] SHARMA, Piyush, et al. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. En Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2018. p. 2556-2565.

[39] ORDONEZ, Vicente; KULKARNI, Girish; BERG, Tamara. Im2text: Describing images using 1 million captioned photographs. Advances in neural information processing systems, 2011, vol. 24.

[40] KRISHNA, Ranjay, et al. Visual genome: Connecting language and Vision using crowdsourced dense image annotations. International journal of computer Vision, 2017, vol. 123, p. 32-73.

[41] GRUBINGER, Michael, et al. The iapr tc-12 benchmark: A new evaluation resource for visual information systems. En International workshop ontoImage. 2006.

[42] Agrawal, Harsh, et al. nocaps: novel object captioning at scale. arXiv:1812.08658, 2019.

[43] PAPINENI, Kishore. BLEU: a method for automatic evaluation of MT. Research Report, Computer Science RC22176 (W0109-022), 2001.

[44] BANERJEE, Satanjeev; LAVIE, Alon. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. En Proceedings of the

acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization. 2005. p. 65-72.

[45] LIN, Chin-Yew. Rouge: A package for automatic evaluation of summaries. En Text summarization branches out. 2004. p. 74-81.

[46] VEDANTAM, Ramakrishna; LAWRENCE ZITNICK, C.; PARIKH, Devi. Cider: Consensus-based image description evaluation. En Proceedings of the IEEE conference on computer Vision and pattern recognition. 2015. p. 4566-4575.

[47] J. Mao, W. Xu, Y. Yang, J. Wang, and A. L. Yuille. Explain images with multimodal recurrent neural networks. arXiv:1410.1090, 2014.

[48] MAO, J. Deep captioning with multimodal recurrent neural networks (m-rnn). arXiv:1412.6632, 2015.

[49] KARPATHY, Andrej; FEI-FEI, Li. Deep visual-semantic alignments for generating image descriptions. En Proceedings of the IEEE conference on computer Vision and pattern recognition. 2015. p. 3128-3137.

[50] KIROS, Ryan; SALAKHUTDINOV, Ruslan; ZEMEL, Rich. Multimodal neural language models. En International conference on machine learning. PMLR, 2014. p. 595-603.

[51] JIA, Xu, et al. Guiding the long-short term memory model for image caption generation. En Proceedings of the IEEE international conference on computer Vision. 2015. p. 2407-2415.

[52] WU, Qi, et al. Image captioning and visual question answering based on attributes and external knowledge. IEEE transactions on pattern analysis and machine intelligence, 2017, vol. 40, no 6, p. 1367-1381.

[53] MUNDU, Albert; SINGH, Satish Kumar; DUBEY, Shiv Ram. ETransCap: efficient *Transformer* for image captioning. Applied Intelligence, 2024, p. 1-15.

[54] STAUDEMEYER, Ralf C.; MORRIS, Eric Rothstein. Understanding LSTM--a tutorial into long short-term memory recurrent neural networks. arXiv:1909.09586, 2019.

[55] ACHARYA, S. "Psychology of Convolutional Neural Network." Towards Data Science, 2019. Disponible en: https://towardsdatascience.com/psychology-of-convolutional-neural-network-53bcd30aac21.

[56] YINGGE, Huo; ALI, Imran; LEE, Kang-Yoon. Deep neural networks on chip-a survey. En 2020 IEEE International Conference on Big Data and Smart Computing (BigComp). IEEE, 2020. p. 589-592.

[57] GUPTA, R. "Convolution Neural Network (CNN) in Deep Learning." Python in Plain English, 2021. Disponible en: https://python.plainenglish.io/convolution-neural-network-cnn-in-deep-learning-77f5ab457166

[58] KAREN, Simonyan. Very deep convolutional networks for large-scale image recognition. arXiv: 1409.1556, 2014.

[59] HE, Kaiming, et al. Deep residual learning for image recognition. En Proceedings of the IEEE conference on computer Vision and pattern recognition. 2016. p. 770-778.

[60] SZEGEDY, Christian, et al. Going deeper with convolutions. En Proceedings of the IEEE conference on computer Vision and pattern recognition. 2015. p. 1-9.

[61] HUANG, Gao, et al. Densely connected convolutional networks. En Proceedings of the IEEE conference on computer Vision and pattern recognition. 2017. p. 4700-4708.

[62] MINGXING, Tan; QUOC, V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946, 2019, vol. 1.

[63] DENG, Jia, et al. Imagenet: A large-scale hierarchical image database. En 2009 IEEE conference on computer Vision and pattern recognition. Ieee, 2009. p. 248-255.

[64] "Introducción a las Redes Neuronales Recurrentes (RNN)." Aprendizaje Profundo. [en línea] Disponible en: https://aprendizajeprofundo.github.io/Libro-Fundamentos/Redes_Recurrentes/Cuadernos/rnn_Intro_Redes_Recurrentes.html [Accedido: 23/06/2024].

[65] BEBIS, George; GEORGIOPOULOS, Michael. Feed-forward neural networks. Ieee Potentials, 1994, vol. 13, no 4, p. 27-31.

[66] JOULIN, Armand, et al. Bag of tricks for efficient text classification. arXiv:1607.01759, 2016.

[67] Mlearning Lab. "Problema de desvanecimiento del gradiente (Vanishing Gradient Problem)." Mlearning Lab, 6 de mayo de 2018. [En línea] Disponible en: https://mlearninglab.com/2018/05/06/problema-de-desvanecimiento-del-gradiente-vanishing-gradient-problem/ [Accedido: 27/08/2024].

[68] Medium. "Image Captioning Using Attention Mechanism." Medium, 2021. [En línea] Disponible en: https://medium.com/swlh/image-captioning-using-attention-mechanism-f3d7fc96eb0e [Accedido: 02/08/2024].

[69] BAHDANAU, Dzmitry. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473, 2014.

[70] TAN, T. "Evolution of Language Models: N-grams, Word Embeddings, Attention & *Transformers*." Towards Data Science, 2023. Disponible en: https://towardsdatascience.com/evolution-of-language-models-n-grams-word-embeddings-attention-*Transformers*-a688151825d2.

[71] LUONG, Minh-Thang. Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025, 2015.

[72] VASWANI, A. Attention is all you need. Advances in Neural Information Processing Systems, 2017

[73] ANDERSON, Peter, et al. Bottom-up and top-down attention for image captioning and visual question answering. En Proceedings of the IEEE conference on computer Vision and pattern recognition. 2018. p. 6077-6086.

[74] Pinecone.io. "Vision *Transformers* for Image Search." Pinecone, 2023. Disponible en: https://www.pinecone.io/learn/series/image-search/vision-*Transformers*/.

[75] RADFORD, Alec, et al. Language models are unsupervised multitask learners. OpenAI blog, 2019, vol. 1, no 8, p. 9.

[76] Dibyansu, D. Flickr 8k Data. Kaggle. Disponible en: https://www.kaggle.com/datasets/dibyansudiptiman/flickr-8k/data. [Accedido: 17/06/2024].

[77] LearnOpenCV. "Understanding Convolutional Neural Networks (CNN)." LearnOpenCV, 2023. Disponible en: https://learnopencv.com/understanding-convolutional-neural-networks-cnn/.

[78] KINGMA, Diederik P. Adam: A method for stochastic optimization. arXiv:1412.6980, 2014.

[79] DotCSV. "¡Redes Neuronales CONVOLUCIONALES! ¿Cómo funcionan?" YouTube, 2020. Disponible en: https://www.youtube.com/watch?v=V8j1oENVz00.

ANEXO ODS

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				x
ODS 2. **Hambre cero.**				x
ODS 3. **Salud y bienestar.**				x
ODS 4. **Educación de calidad.**		x		
ODS 5. **Igualdad de género.**				x
ODS 6. **Agua limpia y saneamiento.**				x
ODS 7. **Energía asequible y no contaminante.**				x
ODS 8. **Trabajo decente y crecimiento económico.**				x
ODS 9. **Industria, innovación e infraestructuras.**		x		
ODS 10. **Reducción de las desigualdades**.			x	
ODS 11. **Ciudades y comunidades sostenibles**.				x
ODS 12. **Producción y consumo responsables.**				x
ODS 13. **Acción por el clima.**				x
ODS 14. **Vida submarina.**				x
ODS 15. **Vida de ecosistemas terrestres.**				x
ODS 16. **Paz, justicia e instituciones sólidas.**			x	
ODS 17. **Alianzas para lograr objetivos.**				x

Reflexión sobre la relación del TFG con los Objetivos de Desarrollo Sostenible.

En este trabajo, se establece una conexión con varios Objetivos de Desarrollo Sostenible (ODS), destacando cómo la generación automática de descripciones de imágenes mediante inteligencia artificial puede contribuir a metas globales clave.

ODS 4: Educación de Calidad

Este TFG contribuye al avance del conocimiento en inteligencia artificial y al desarrollo de técnicas avanzadas en generación de descripciones de imágenes. Estas innovaciones pueden ser incorporadas en entornos educativos, proporcionando herramientas valiosas para la enseñanza de la tecnología y la ciencia de datos. Además, facilita la comprensión de conceptos complejos de IA, promoviendo una educación de calidad en el ámbito de la inteligencia artificial y la computación.

ODS 9: Industria, Innovación e Infraestructuras

El desarrollo de modelos de generación automática de descripciones de imágenes, como los presentados en este trabajo, fomenta la innovación tecnológica. Al crear y optimizar modelos de IA, se impulsa el avance tecnológico.

ODS 10: Reducción de las Desigualdades

La tecnología de generación automática de descripciones tiene el potencial de hacer la información más accesible para personas con discapacidades, especialmente para aquellos con discapacidad visual. Al proporcionar descripciones detalladas y contextuales de imágenes, esta tecnología puede mejorar la accesibilidad y la inclusión, reduciendo las barreras de comunicación.

ODS 16: Paz, Justicia e Instituciones Sólidas

Los modelos de inteligencia artificial utilizados para la generación de descripciones de imágenes también tienen aplicaciones en contextos de seguridad y análisis de datos. Por ejemplo, estos modelos pueden ser empleados en sistemas de vigilancia para interpretar imágenes de manera autónoma.

Estos vínculos con los ODS subrayan cómo la investigación y el desarrollo en inteligencia artificial pueden contribuir significativamente a diversos aspectos del desarrollo sostenible, ofreciendo beneficios tanto en el ámbito educativo como en la mejora de la infraestructura, la reducción de desigualdades y la promoción de la paz y la justicia.