UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DPTO. DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

MÁSTER UNIVERSITARIO EN INTELIGENCIA ARTIFICIAL, RECONOCIMIENTO DE FORMAS E IMAGEN DIGITAL

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

"Descripción automática de imágenes"

TRABAJO DE FIN DE MÁSTER

Autor/a:
Pablo Pallarés Font de Mora

Tutor/a:
Francisco Casacuberta Nolla
Roberto Paredes Palacios

CURSO: 2020-2021

Resumen

El propósito de este Trabajo es el estudio, implementación, y desarrollo de sistemas, basados en Deep Learning, orientados a la generación automática de descripciones de imágenes o Image Captioning. Este campo aúna las áreas del Procesamiento del Lenguaje Natural (PLN), y de la Visión por Computador (VPC). Antes proceder a la implementación, se ha realizado un análisis de los diferentes enfoques utilizados para abordar esta tarea, los corpus disponibles formato: [Imagen - Caption/s], y las arquitecturas o modelos utilizados. Tras este análisis, se ha optado, inicialmente, por abordarlo desde el enfoque más usual: basado en modelos del lenguaje, con una arquitectura Codificador-Decodificador. Para ello, se realiza una codificación de: las descripciones (captions) a un espacio vectorial de Embeddings Word2Vec, por una parte, y, por otra, las imágenes haciendo uso redes convolucionales CNN. Con esta información codificada, el Decodificador es el encargado de aprender un Modelo del Lenguaje con redes neuronales recurrentes RNN capaz de Generar descripciones. Las diferentes implementaciones de este trabajo se han realizado bajo la plataforma de software Python, empleando la biblioteca de código abierto TensorFlow, orientada al entrenamiento de modelos de Aprendizaje automático, y el framework de alto nivel para el aprendizaje, Keras.

Palabras Clave

Descripción de Imágenes; Procesamiento del Lenguaje Natural; Visión por Computador; arquitectura Codificador-Decodificador; Aprendizaje automático.

Resum

El propòsit d'aquest Treball és l'estudi, implementació, i desenvolupament de sistemes, basats en Deep Learning, orientats a la generació automàtica de descripcions d'imatges o Image Captioning. Aquest camp conjumina les àrees del Processament del Llenguatge Natural (PLN), i de la Visió per Computador (VPC). Abans procedir a la implementació, s'ha realitzat un anàlisi dels diferents enfocaments utilitzats per a abordar aquesta tasca, els corpus disponibles, en format: [Imatge - Caption/s], i les arquitectures o models utilitzats. Després d'aquest anàlisi, s'ha optat, inicialment, per abordar-lo des de l'enfocament més usual: basat en models del llenguatge, amb una arquitectura Codificador-Decodificador. Per a això, es realitza una codificació de: les descripcions (captions) a un espai vectorial de Embeddings Word2Vec, d'una banda, i, per una altra, les imatges fent us de xarxes convolucionals CNN. Amb aquesta informació codificada, el Decoder és l'encarregat d'aprendre un Model del Llenguatge amb xarxes neuronals recurrents RNN capaç de Generar descripcions. Les diferents implementacions d'aquest treball s'han realitzat sota la plataforma de programació Python, emprant la biblioteca de codi obert TensorFlow, orientada a l'entrenament de models d'Aprenentatge automàtic, i el framework d'alt nivell per a l'aprenentatge, Keras.

Paraules clau

Descripció d'imatges; Processament del Llenguatge Natural; Visió per Computador; arquitectura Codificador-Decodificador; Aprenentatge automàtic.

Abstract

The purpose of this work is the study, implementation, and development of Deep Learning systems, oriented to Image Captioning. This field combines the areas of Natural Language Processing (NLP) and Computer Vision (CV). Before proceeding to the implementation, an analysis of the different approaches used to tackle this task has been carried out, the available corpora with format: [Image - Caption/s], and the architectures or models used. After this analysis, it has been chosen, initially, to approach it from the most usual approach: based on language models, with an Encoder-Decoder architecture. For this purpose, the descriptions (captions) are encoded in a vector space of Word2Vec Embeddings on the one hand, and on the other hand, the images using CNN convolutional networks. With this encoded information, the Decoder is in charge of learning a Language Model with Recurrent Neural Network RNN capable of generating descriptions. The different implementations of this work have been carried out under the Python software platform, using the open-source library TensorFlow, oriented to the training of Machine Learning models, and the high-level framework for learning, Keras.

Keywords

Image Captioning; Natural Language Processing; Computer Vision; Encoder-Decoder architecture; Machine Learning.

Índice

Tabla de ilustraciones

Tabla de Tablas

1. Introducción

1.1. Presentación

Hoy en día tenemos a nuestro alcance una gran cantidad de información a través de internet, donde cada vez se imponen más los contenidos multimedia: videos o imágenes. Desgraciadamente, este tipo de información no está disponible para un cierto sector de la población que, o bien, padece deficiencias visuales, o, simplemente, el peso del contenido es prohibitivo para las velocidades de internet de las que disponen. La subtitulación de imágenes manual generada por los proveedores de estos contenidos puede suponer un alivio a este problema. Consiste en dada una imagen, generar una descripción que sea lingüísticamente plausible y semánticamente adecuada al contenido de la imagen. En el caso de las personas con deficiencias visuales, sería necesaria, además, la utilización de sistemas **Text-To-Speech** [1] para la locución de estas descripciones del contenido.

La descripción de ilustraciones es innata para el ser humano y abre un amplio abanico de posibilidades, además de las ya mencionas [2]. Entre estas posibilidades, podríamos destacar: la búsqueda web de archivos gráficos, la extensión a imágenes dinámicas (Videos) para su análisis o extracción de información, por ejemplo, para la descripción escénica de una película. También podría ser interesante de cara al enriquecimiento gramatical y lingüístico en tareas de indexación de imágenes para uso medico, docente… con el principal objetivo de facilitar la recuperación de información (visual) y permitir búsquedas más complejas. Sin embargo, esta tarea de subtitulación puede resultar ardua y tediosa para los proveedores de contenido, y, por tanto, omitida en muchas ocasiones. Un amplio porcentaje de material audiovisual de la web carece de subtítulos, y precisamente por este motivo, los sistemas de generación automática de subtítulos de imágenes, más comúnmente conocidos por la comunidad científica como: **Automatic Image Captioning systems** han tomado importancia en los últimos años. [3] [4]

Esta tarea aúna el área de la Visión por Computador **(CV)**, encargada de enfrentar el problema de la compresión y extracción de características de una imagen, con el área del Procesamiento del Lenguaje Natural **(NLP)**, para resolver el problema de la generación de una descripción coherente y lingüísticamente correcta. Típicamente este problema se había plantado desde 3 enfoques diferentes: **Template-based methods**, **Search-based Methods**, y en el que se va a centrar especialmente este trabajo **Language-based Methods,** basado en modelos del lenguaje y Arquitecturas Encoder-Decoder. [5]

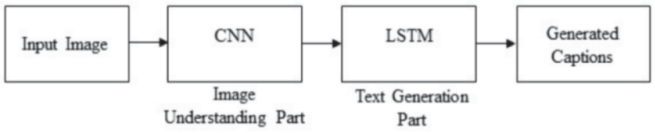

Ilustración 1: Arquitectura general Encoder-Decoder [4]

No obstante, uno de los principales problemas de estos enfoques clásicos es que tienden a replicar frases vistas en el conjunto de entrenamiento, y la mayoría de los subtítulos generados siguen los mismos patrones. Algunos de los trabajos recientes pretenden aportar mayor diversidad en las descripciones utilizando las redes conocidas como Generative Adversarial Networks (GAN) [6] [7], sin embargo, este tipo de técnicas sacrifican el rendimiento general en los criterios de evaluación estándar. Otro trabajo [8], plantea generar subtítulos discriminativos para una imagen en el contexto de otras

imágenes semánticamente similares, pero requiere en inferencia tanto la imagen que se desea describir, como en imágenes distractoras. Otros trabajos destacados en este marco de estudio introducen algoritmos Reinforce [5], donde es necesario recurrir a una tarea complementaria conocida como: **Image Retrieval.** Es la tarea "inversa" a Image Captioning, consiste en generar una imagen a partir de una o varias descripciones.

1.2. Motivación y Objetivos

La motivación principal de este trabajo es establecer un entorno de trabajo que permita estudiar, y comprender el comportamiento teórico y práctico de los sistemas utilizados para la generación automática de descripciones de imágenes "SISTEMAS IMAGE CAPTIONING". Los objetivos pretendidos en este proyecto son los siguientes:

- Estudiar los principios del Reconocimiento de Formas, y los sistemas donde se aplica.
- Estudiar el funcionamiento y comportamiento general de los sistemas Image Captioning.
- Revisar y documentar del estado del Arte de estos sistemas en relación con las Arquitecturas utilizadas, los Corpus, las métricas de evaluación, y resultados obtenidos en trabajos similares.
- Decidir las arquitecturas, corpus y métricas de evaluación a utilizar en el proyecto.
- Descargar los datos, establecer una topología (Modelo Neuronal) y realizar una partición de los datos para dividirlos en conjunto de entrenamiento, conjunto de validación y conjunto de prueba.
- Adaptar los datos, es decir, extraer y preprocesar las imágenes, y el texto de las descripciones, etc.
- Implementar la topología seleccionada, y realizar el entrenamiento de los pesos de las partes del modelo neuronal que requieran entrenamiento.
- Realizar pruebas con el/los modelo/s entrenado/s en fase inferencia para evaluar los resultados de la generación de descripciones el con las métricas seleccionadas.
- Analizar los resultados y compararlos con los resultados estado del Arte.
- Implementar mejoras en el caso de que se considere oportuno, y probar nuevas topologías.

1.3. Metodología

Al tratarse de un proyecto académico, principalmente orientado a la investigación, sigue la metodología común para este tipo de proyectos. Se parte de un extenso estudio del estado del arte para la comprensión del problema. Seguidamente, se establece un marco de trabajo donde se propone un modelo para resolver el problema objetivo utilizando un conjunto de datos determinado, el cual también hay que comprender, se realiza una extensa experimentación, para ajustar el modelo propuesto y, por último, se evalúan los resultados.

Tal y como se ha mencionado para poder establecer el entorno de trabajo, es necesaria la realización de un estudio de campo donde, se revisarán las diferentes líneas de investigación seguidas en este área durante los últimos años, es decir, se hará un análisis del "Estado del Arte" para poder establecer un modelo propio. Durante este estudio de campo, también se pretende explorar los diferentes conjuntos de datos disponibles para esta tarea, y determinar cual o cuales utilizar en nuestro trabajo.

Para el desarrollo e implementación de este trabajo se van a utilizar las herramientas TensorFlow y Keras. Aprovechando los conocimientos adquiridos durante el máster de Keras se pretende importar y utilizar modelos pre-entrenados para el codificador, y Word Embeddings también pre-entrenados, para agilizar y facilitar la tarea de ajuste del Decodificador.

Por ultimo, se realizará una evaluación a través de métricas propias de traducción Automática o generación de resúmenes, como pueden: BLEU [9], METEOR [10], ROUGE [11], CIDEr [12], SPICE [13]… debido a que la tarea en cuestión en cierta medida realiza una traducción de imagen a Texto. Estás métricas nos servirán de orientación para saber si se han alcanzado los resultados estado del Arte.

Una vez Evaluado el modelo, se comparan los resultados, con los de otros trabajos similares, para comprobar la calidad del modelo propuesto, y poder continuar con el ajuste del modelo, o dar por finalizado el proyecto. La metodología descrita en este apartado esta inspirada en el modelo de análisis de datos ampliamente conocido y adoptado CRISP-DM (Cross-Industry Standard Process for Data Mining) [14].

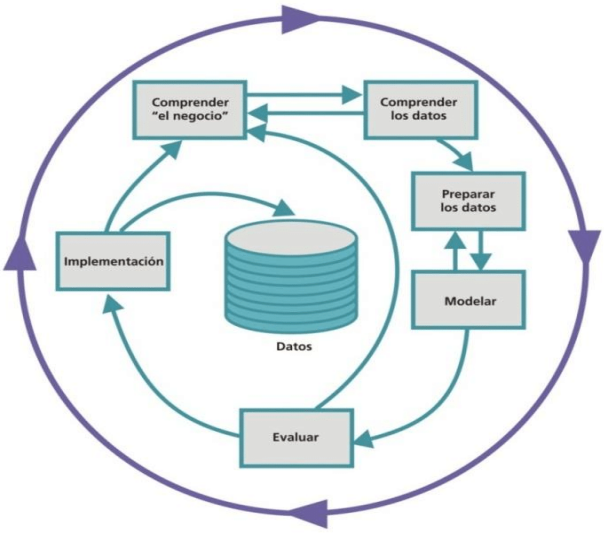

Ilustración 2: Modelo CRISP-DM adaptado por Yanina Noemí Bellini Saibene [15]

1.4. Organización de la Memoria

El informe tiene la siguiente estructura:

- **Capítulo 1: Introducción.** En este capítulo se realiza una presentación del proyecto, así como las motivaciones que han conducido al desarrollo de este trabajo, y los objetivos a cumplir. También se detalla la estructura seguida en este documento y la metodología empleada durante el desarrollo.

- **Capítulo 2: Estado del Arte.** Durante este capítulo se repasan los diferentes enfoques, conjuntos de datos, arquitecturas, métricas utilizadas, y resultados obtenidos en otros trabajos previos que abordan esta tarea.

- **Capitulo 3: Redes neuronales para el tratamiento de secuencias: la arquitectura Codificador-Decodificador.** Se realiza una descripción de la arquitectura Encoder-Decoder, y su aplicación al problema de descripción automática de imágenes

- **Capítulo 4: Marco Experimental.** Reúne la parte más importante del trabajo. Se describen el conjunto de datos, y su partición, las diferentes topologías con las que se va a experimentar, así como el software utilizado y el ajuste de hiperparámetros para la consecución de los resultados deseados.

- **Capítulo 5: Resultados experimentales y análisis.** Incluye el análisis de los resultados obtenidos, y ejemplos de funcionamiento. Se extraen conclusiones en relación a las topologías propuestas, y se presentan los principales problemas encontrados.

- **Capítulo 6: Conclusiones y trabajos futuros.** Se exponen las conclusiones alcanzadas realizar el proyecto. También se hace un breve repaso de todos los puntos vistos en la memoria de forma objetiva y concisa. Además, se plantean nuevas ideas o posibilidades que podrían incorporarse en un futuro trabajo con el fin de mejorar el actual.

- **Bibliografía.** Detalla las fuentes de información utilizadas durante la elaboración del proyecto.

- **Declaración de trabajo original.** Se reconoce el trabajo como propio y exento de copia a terceros.

1.5. Principales problemas

Los principales inconvenientes encontrados vienen dados por diversos factores:

- En primer lugar, la complejidad de reunir toda la información, y estructurarla para componer un estado del arte que permitiera tener una visión general del problema a abordar.

- El segundo de los problemas, la comprensión del dataset seleccionado y establecer un sistema de lectura de datos que no sobrepasara la memoria de los equipos utilizados.

- Y el último, y principal problema, ha sido establecer el marco experimental que permitiera el entrenamiento de los modelos neuronales, debido a que se ha trabajado con modelos pesados (RNN), y se ha optado por un método de entrenamiento conocido como "**Teacher Forcing for Recurrent Neural Networks**" [16]. Explicado en capítulo 4 de este informe.

2. Estado del Arte

En primer lugar, es imprescindible la realización de un estudio de campo para conocer los diferentes enfoques, corpus, arquitecturas y métricas utilizadas en otros trabajos que abordan esta tarea, así como, los resultados obtenidos en estos trabajos previos. Tal y como se anticipaba en la introducción hay 3 enfoques clásicos utilizados para abordar esta tarea:

- **Template-based methods** [17] [18] [19]**:** consiste en la generación de subtítulos haciendo uso de plantillas lingüísticas.

- **Search-based methods** [20]**:** se basa en la búsqueda de un subtítulo adecuado dentro de un conjunto de subtítulos de imágenes semánticamente similares.

- **Language-based methods:** es el enfoque más habitual en los trabajos recientes. Se pueden dividir en dos categorías principales: un método basado en un modelos de lenguaje de probabilidad estadística para generar características artesanales [21] y, en segundo lugar, un método de aprendizaje profundo basado en un modelo de lenguaje con arquitecturas Codificador-Decodificador para extraer características profundas. Los detalles específicos de los dos modelos se discutirán por separado.

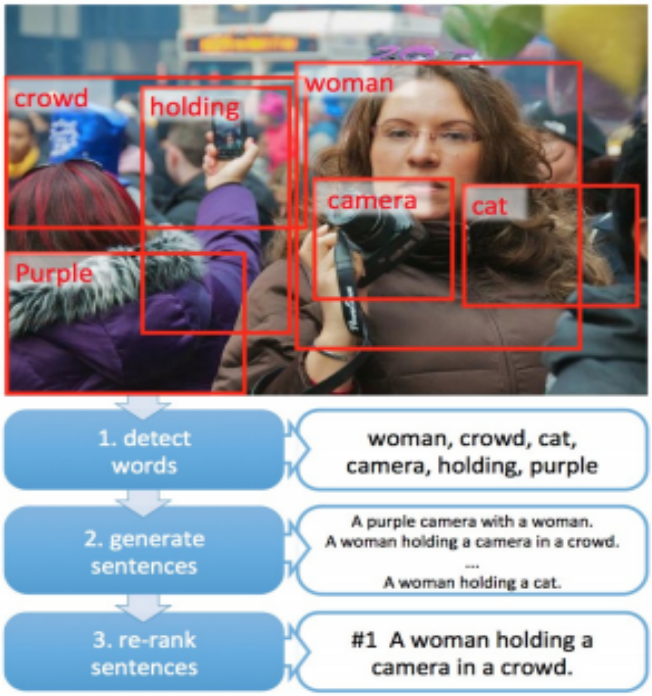

Ilustración 3: Método basado en el modelo de lenguaje y detector visual [21]

En la ilustración 3 se muestra el funcionamiento de la primera de las categorías descritas en el enfoque de "Language-based methods". Se trata de un sistema Midge basado en la estimación de máxima verosimilitud, que aprende directamente el detector visual y el modelo de lenguaje a partir del conjunto de datos de descripción de imágenes. Tal y como se muestra en la ilustración. Fang et al. [21] analizan primero la imagen, detectan el objeto y generan un pie de foto. Las palabras se detectan aplicando una red neuronal convolucional (CNN) al área de la imagen e integrando la información con MIL [22].

A continuación, se entrena la estructura de la frase directamente a partir del pie de foto para minimizar las suposiciones a priori sobre la estructura de la frase. Finalmente, convierte un problema de generación de subtítulos de imágenes un problema de optimización y busca la frase más probable.

Sin embargo, como ya se ha mencionado, el principal problema de los 3 enfoques mencionados es la tendencia a replicar descripciones. Ha habido diferentes propuestas para solventarlo: como las redes GAN [6] [7], o imágenes distractoras [8], pero casi todas añaden nuevos problemas. Los trabajos más competitivos a nivel de resultados parten del enfoque de modelos del lenguaje con Deep Learning, pero trabajan con el denominado Aprendizaje por refuerzo (Reinforcement Learning) [5], donde se añade el nuevo modulo de **Image Retrieval**. Con ambos módulos: **Captioning Module** (genera subtítulos de imágenes dadas) y **Self-retrieval Module** (genera imágenes a partir de subtítulos y actúa como evaluador para medir la calidad de los subtítulos y alentar al modelo a generar subtítulos discriminativos), y adoptando el algoritmo REINFORCE para calcular gradientes se consiguen resultados estado del arte muy competitivos hasta la fecha.

2.1. Arquitecturas

En este apartado se muestran algunas de las técnicas más relevantes en la línea de investigación de Image Captioning. Con ello, se pretende justificar las topologías escogidas en este trabajo para la consecución de los objetivos propuestos. En la Tabla 1, se recogen algunas de las arquitecturas utilizadas en diferentes publicaciones estado del arte. En este trabajo vamos a centrarnos principalmente en el estudio del enfoque de los "Language-based Methods" con arquitecturas Codificador-Decodificador. Las Redes Convolucionales (CNN) y las Redes Recurrentes (RNN) son la cabeza y la cola de estas arquitecturas.

CODIFICADOR

Las redes neuronales convolucionales (CNN) han sido, y siguen siendo las más utilizadas para la codificación y extracción de características de las imágenes. Estas redes consiguieron resolver el problema del *Representation Learning*: aprender una representación adecuada de los objetos a clasificar en una imagen (objetos complejos). Además de en tareas de clasificación de imágenes, también se pueden utilizar en otro tipo de tareas como, por ejemplo:

- **Detección de objetos**: Se ajustan los pesos utilizando como target uno o varios Bounding Box (Cuadro delimitador) que indica la posición de los objetos en la imagen de entrada.

- **Segmentación de la imagen**: En este caso en lugar de indicar la posición de los objetos con un cuadro delimitador, se afina más y se indica cada pixel de la imagen a que clase pertenece (Barco, césped, lago, etc.).

En cualquier caso, el objetivo final es aprender los pesos de la Red Neuronal en entrenamiento utilizando un conjunto de datos adecuado para la tarea en cuestión. En los últimos años las CNN han ido evolucionando, algunas de las topologías más relevantes han sido: AlexNet, GoogleNet, VGG, ResidualNet, DenseNet, etc.

DECODIFICADOR

Las redes neuronales recurrentes (RNN) [23] han atraído mucha atención en el campo del aprendizaje profundo. Originalmente fueron ampliamente utilizadas en el campo del procesamiento del lenguaje natural, logrando buenos resultados en el modelado del lenguaje [24]. En el campo del habla, las RNN se encargan de convertir el texto y el habla entre sí [25] [26] . También han sido utilizadas en el área de la traducción automática [27] [28], la sesión de preguntas y respuestas [29] [30], etc. Por supuesto, también se utilizan como potentes modelos lingüísticos a nivel de caracteres y palabras. Actualmente, los modelos a nivel de palabras parecen ser mejores que los modelos a nivel de caracteres, pero esto es únicamente circunstancial. Las RNN también están ganando rápidamente popularidad en la visión por ordenador. Por ejemplo, la clasificación de vídeos a nivel de fotogramas [31] [32] [33], el modelado de secuencias [34] [35] y las recientes tareas visuales de pregunta-respuesta. En este caso, el método de generación de descripciones de imágenes basado en el modelo codificador-decodificador se propone con el auge y la aplicación generalizada de la red neuronal recurrente [36]. Las RNN, como pueden ser las Long Short Term Memory (LSTM) [37] descritas en el siguiente capítulo, o las Gated Recurrent Unit (GRU) [38], tienen la función de llevar a cabo la decodificación de estas características extraídas por el codificador en frases, o en este caso concreto en la descripción de las imágenes en cuestión.

Publicación	Codificación imágenes	Decodificador
Karpathy et al. (2014)	AlexNet	DTR
Kiros et al. (2014a)	AlexNet	LBL
Kiros et al. (2014b)	AlexNet, VGG	LSTM,SC-NLM
Mao et al. (2014)	AlexNet	RNN
Chen and Zitnick (2015)	VGG	RNN
Donahue et al. (2015)	Caffe	LSTM
Devlin et al. (2015)	VGG	MELM
Fang et al. (2015)	AlexNet,VGG	MELM
Jia et al. (2015)	VGG	LSTM
Karpathy and Fei-Fei (2015)	VGG	RNN
Ma et al. (2015)	VGG	LCNN
Mao and Yuille (2015)	AlexNet,VGG	RNN
Vinyals et al. (2015)	Inception-V1	LSTM
Xu et al. (2015)	AlexNet	LSTM
Hendricks et al. (2016)	VGG	LSTM
Johnson et al. (2016)	VGG	LSTM
Ma and Han (2016)	AlexNet	LSTM
Mao et al. (2016)	AlexNet,VGG	LSTM
Mathews et al. (2016)	Inception-V1	LSTM
Sugano and Bulling (2016)	VGG	LSTM
Tran et al. (2016)	ResNet	MELM
Wang et al. (2016)	VGG	LSTM
Yang et al. (2016)	VGG	LSTM
You et al. (2016)	Inception-V1	RNN
Chen et al. (2017a)	VGG,ResNet	LSTM
Dai et al. (2017)	VGG	LSTM
Fu et al. (2017)	VGG	LSTM

Gan et al. (2017a)	ResNet	LSTM
Gan et al. (2017b)	ResNet	LSTM
Gu et al. (2017)	VGG	LCNN,LSTM
Liu et al. (2017a)	VGG	LSTM
Liu et al. (2017c)	Inception-V3	LSTM
Liu et al. (2017b)	CNN+LSTM	LSTM
Lu et al. (2017)	ResNet	LSTM
Niu et al. (2017)	AlexNet	HM-LSTM
Park et al. (2017)	ResNet	LSTM
Pedersoli et al. (2017)	VGG	RNN
Rennie et al. (2017)	ResNet	LSTM
Shetty et al. (2017)	Inception-V1	LSTM
Tavakoliy et al. (2017)	VGG	LSTM
Venugopalan et al. (2017)	VGG	LSTM
Wang et al. (2017)	ResNet	LSTM
Yang et al. (2017)	VGG	LSTM
Yao et al. (2017b)	Inception-V1	LSTM
Yao et al. (2017a)	VGG	LSTM
Zhang et al. (2017)	Inception-V3	LSTM
Aneja et al. (2018)	VGG	LCNN
Jiang et al. (2018)	Inception-V3	LSTM
Khademi and Schulte (2018)	VGG,ResNet	Grid LSTM
Li and Chen (2018)	Faster R-CNN	LSTM
Wang and Chan (2018)	VGG	LCNN

Tabla 1: Resumen Arquitecturas Codificador-Decodificador usadas en trabajos estado del arte

En la tabla 1 es evidente, que las topologías utilizadas en los trabajos presentados han sido cambiantes en función de las topologías estado del arte en cada campo en el periodo de la publicación. Tanto las CNN utilizadas como Codificador de imágenes, donde podemos encontrar topologías famosas como: Alexnet, VGG, Inception, Resnet, etc. También las topologías utilizadas como Decodificador, donde las más habituales son las redes recurrentes como las LSTM o las GRU. Últimamente, la inclusión de mecanismos de atención a las RNN para seleccionar mejor la información útil a la hora de generar recursivamente las diferentes palabras que componen una descripción, y otras topologías que utilizan mecanismos de Multi-Atención como es el caso de los Transformers, como Bert están copando los trabajos del área del PLN, incluso empiezan a introducirse con buenos resultados en el área de la VPC.

2.2. Datasets

Los datos son la base de la inteligencia artificial. A menudo la ciencia se encuentra con la complejidad de descifrar leyes difíciles de encontrar, y es a partir de una gran cantidad de datos la mejor forma de encontrarlas. En la tarea de generación de descripciones de imágenes, actualmente existen conjuntos como: MSCOCO, Flickr8k, Flickr30k, PASCAL 1K, IAPTR-TC12, Abstract Scenes, Conceptual Captions, VLT2K, etc. La mayoría de los datasets que se utilizan para la tarea de Image Captioning están orientados, también, a una tarea estrechamente relacionada, como es la detección de objetos en una imagen. Por eso, además de incluir imágenes y descripciones, también tienen boundig box, clases, etc. En la tabla 1 ya se podían apreciar diferentes topologías propias de el problema de Object detection como Faster R-CNN.

Conjunto de datos	Imágenes	Textos	Objetos
Pascal1K (Rashtchian et al., 2010) [39]	1K	5	Partial
IAPTR-TC12 (Escalante et al., 2010) [40]	20K	1-5	Segmented
VLT2K (Elliott and Keller, 2013)	2K	3	Partial
Abstract Scenes (Zitnick et al., 2013)	10K	6	Complete
Flickr8K (Hodoshetal.,2013) [41]	8K	5	No
Flickr30K (Young et al., 2014) [42]	31K	5	No
MSCOCO (Lin et al., 2014) [43]	328K	5	Partial
Flickr30K Entities (Bryan P. et al., 2016)	31K	5	Partial
Conceptual Captions (Sharma et al., 2018)	3.3M	1	No

Tabla 2: Resumen de los datasets de imágenes más conocidos

De entre los distintos corpus presentados en la Tabla 2, los más utilizados para esta tarea son el Pascal 1k, el conjunto Flickr8k y su versión extendida Flickr30k, y especialmente MSCOCO de Microsoft.

- ***MSCOCO*** [43]. El conjunto de datos de Microsoft COCO Captions fue desarrollado por el equipo de Microsoft, y tiene como objetivo la comprensión de la escena. Captura imágenes de escenas diarias complejas y se puede utilizar para realizar múltiples tareas como el reconocimiento, la segmentación y la descripción de imágenes. El conjunto de datos utiliza el servicio "Mechanical Turk" de Amazon para generar artificialmente al menos cinco descripciones para cada imagen, con un total de más de 1,5 millones de descripciones. La partición del conjunto se suele hacer en un conjunto de entrenamiento que contiene 82.783 imágenes, el conjunto de validación tiene 40.504 imágenes y el conjunto de prueba tiene 40.775 imágenes. Su versión de 2014 de los datos tiene un total de aproximadamente 20G imágenes y aproximadamente 500M de archivos de anotaciones que marcan la correspondencia entre una imagen y sus descripciones.

- ***Flickr8k / Flickr30k*** [41] [42]. Las imágenes del conjunto Flickr8k provienen del sitio web de álbumes de fotos de Yahoo, Flickr, que contiene 8.000 fotos, típicamente de 6000 imágenes de entrenamiento, 1000 imágenes para validación y 1000 imágenes de prueba. Flickr30k es la versión extendida del anterior, y contiene 31,783 imágenes recopiladas del sitio web de Flickr, nuevamente suelen ser capturas de escenas del mundo real, y contienen 5 descripciones por cada imagen

- ***PASCAL 1K*** [39]. Se trata de un subconjunto del famoso conjunto de datos de imágenes de desafío PASCAL VOC, que proporciona un conjunto de datos de anotación de imágenes estándar y un sistema de evaluación estándar. La colección de fotografías de PASCAL VOC consta de 20 clases. Para cada una de sus 20 clases, se seleccionaron 50 imágenes al azar, obteniendo un conjunto total de 1,000 imágenes. Luego, el servicio de robot turco de Amazon se utiliza para marcar manualmente cinco descripciones para cada imagen. La calidad de la imagen del conjunto de datos es buena y la etiqueta está completa, lo cual es muy adecuado para probar el rendimiento del algoritmo.

Finalmente, se ha decidido trabajar con Flickr30k, por diversos motivos: por simplicidad de montaje, y por tener un tamaño suficientemente extenso para la magnitud de proyecto que se pretende abordar (ni es tan reducido como el Pascal o el flickr8k, ni se extiende hasta 330 mil como el de Microsoft).

2.3. Métricas de Evaluación

La tarea de descripción de imágenes es similar a la traducción automática, y su método de evaluación se extiende desde la traducción automática para formar sus propios criterios de evaluación únicos. Para evaluar un modelo en función de la calidad de las descripciones generadas en el lenguaje utilizado, se requiere utilizar una serie de métricas particulares, ya que métricas clásicas como accuracy, precisión o recall no son totalmente representativas de la calidad del sistema a la hora de comparar dos textos en lenguaje natural. Por ello, en esta tarea se han utilizado una serie de estándares, originales de la traducción automática, o el resumen automático de textos, para realizar una comparativa entre las descripciones esperadas y las descripciones generadas.

- **BLEU (BiLingual Evaluation Understudy)** [9]**:** es la métrica más utilizada en la práctica. El propósito original de esta métrica no es el problema de la descripción de imágenes, sino el problema de la traducción automática. Se basa en la evaluación de la tasa de precisión. Se utiliza para analizar la correlación de n-grama coincidentes entre la traducción generada por el sistema y el enunciado de traducción de referencia. Consiste en detectar la cantidad de palabras individuales que coinciden, 2-gramas, 3-gramas, y 4-gramas. Se obtiene la media geométrica. (BLEU-1, BLEU-2, BLEU-3, BLEU-4). Un n-grama es una subsecuencia de una secuencia dada. Por ejemplo, para una frase dada "Un perro corriendo sobre la hierba", sus unigramas serían:

UNIGRAMAS					
Un	perro	corriendo	sobre	la	hierba

En cambio, la lista de bigramas o 2-gramas sería:

Bigramas				
Un perro	Perro corriendo	Corriendo sobre	sobre la	la hierba

Tal y como se ha mencionado, a partir de los n-gramas tenidos en cuenta (1, 2, 3, y 4) se determina la precisión de las descripciones candidatas obtenidas por el sistema. La precisión se entiende como el ratio de n-gramas presentes en el candidato respecto al total de n-gramas presentes en la frase de referencia:

$$Precisión_n = \frac{c(\,\delta(nc,\ nr)\,)}{c(\,nc\,)}$$

En esta función ***nc***, y ***nr*** son los n-gramas de la descripción candidata y referencia respectivamente. La expresión $\boldsymbol{\delta(X,Y)} = 1$ cuando X e Y son iguales, y 0 cuando son diferentes. Por otra parte, ***c(X)*** es el número total de eventos que se producen, es decir que se realiza el conteo revisando los n-gramas de la descripción candidata y la referencia comunes en el numerador, y un conteos de los n-gramas de la descripción candidata en el denominador. Véase un pequeño ejemplo:

Referencia: Un perro corriendo sobre la hierba.
Candidato: Un perro perro sobre la hierba

La precisión$_1$ del candidato sería: **(1+1+1+1+1) /6 = 6/6= 1**. La precisión no es una medida adecuada para calcular la similitud entre 2 frases, como se ha podido observar. Por ello, es interesante tener en cuenta el número máximo de ocurrencias de un n-grama en la frase de referencia, siendo éste el límite a la hora de contabilizar las apariciones en la frase candidata. Con esto en cuenta la precisión modificada sería de 5/6 en este ejemplo.

También se puede realizar una penalización por brevedad. Si las frases a comparar tienen una longitud muy distinta, no podemos afirmar que sean similares. En el caso de que la frase candidata tiene mayor longitud que la de referencia, este aspecto se ve reflejado en la fórmula de precisión modificada anterior. Habrá muchos n-gramas en la frase candidata que no aparecerán en la frase de referencia por lo que la precisión será menor. Esto no ocurre cuando la frase candidata es mucho menor, tal y como se puede observar en el siguiente ejemplo:

Referencia: Un perro corriendo sobre la hierba.
Candidato: Un un.

La precisión modificada sería de 2/2=1, y no reflejaría la similitud entre ambas frases. Se introduce un penalizador por brevedad (PB) de las frases candidatas.

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

donde c es la longitud de la frase candidata y r la longitud de la frase de referencia.

Una vez comprendido el cálculo de la precisión para n-gramas, y tal y como se anticipaba con anterioridad, cálculo de BLEU se utiliza la media geométrica para los N n-gramas mencionados. Cada n-grama tendrá un peso ω_n tal que $\sum_{n=1}^{N} \omega_n$. Típicamente $\omega_n = \frac{1}{N} = 1$

La idea central de esta métrica es que cuanto más cerca esté el enunciado generado mediante la traducción automática y el enunciado generado por un profesional humano de la traducción, mejor será el rendimiento del sistema. En esta tarea, el procesamiento es el mismo que la traducción automática: cada imagen equivale a una oración en el idioma de origen en la traducción. La ventaja de BLEU es que la granularidad que considera es un n-grama en lugar de una palabra, considerando una información de coincidencia más larga. La desventaja de BLEU es que, independientemente del tipo de n-grama con el que se coincida, se tratará igual. Por ejemplo, la importancia de la coincidencia del verbo debería ser intuitivamente mayor que la del artículo. Cuanto mayor sea la puntuación BLEU, mejor será el rendimiento.

- **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)** [11]**:** es un conjunto de métricas utilizado habitualmente para evaluar tareas de resumen automático y traducción automática. Nuevamente, se basa en la comparación de n-gramas entre una hipótesis frente a una o varias referencias. Esta métrica tiene numerosas variantes, una de ellas es ROUGE-N Compara los n- gramas (normalmente entre 1 y 3) con los de referencia siendo el ratio de n-gramas comunes dividido por el total de n-gramas en el texto de referencia. Otra variante es ROUGE-L, la cual mide la subsecuencia de palabras coincidentes más larga entre la frase generada y la tomada como referencia. Una subsecuencia de palabras no se debe no requiere de coincidencias consecutivas, como en el caso de un Substring. BLEU y ROUGE son compatibles ya que la primera adopta el rol de precisión mientras que la segundo lo hace de recall. Cuanto más alta sea la puntuación ROUGE, mejor será el rendimiento

- **METEOR (Metric for Evaluation of Translation with Explicit ORdering)** [10]: también es una métrica que se utiliza para la evaluación del resultado de la traducción automática. Al igual que BLEU, la unidad básica de evaluación es la oración, el algoritmo primero crea un alineamiento entre la traducción generada a partir del modelo de traducción automática con el enunciado de la traducción de referencia. Véase un ejemplo:

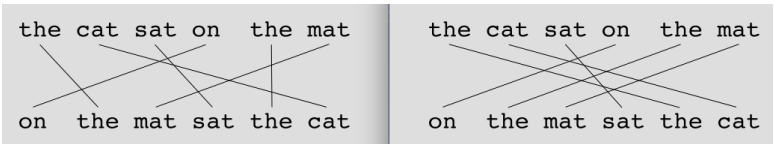

Ilustración 4: alineamiento (a) y alineamiento (b)

El alineamiento se puede entender como el mapeo con una línea entre un unigrama en una cadena y un unigrama en otra cadena. Las restricciones son las siguientes; cada unigrama en la descripción candidata debe correlacionarse con cero o un unigrama en la referencia. Si hay dos alineamientos con el mismo número de asignaciones, el alineamiento con menor cantidad de cruces. De los dos alineamientos mostrados, el alineamiento (a) se seleccionaría en este punto. Las etapas se ejecutan consecutivamente y cada etapa solo agrega a la alineación aquellos unigramas que no se han emparejado en etapas anteriores. Una vez que se calcula el alineamiento final, la puntuación de esta métrica se calcula de la siguiente forma: La precisión de unigrama P se calcula como:

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

Donde m es el número de unigramas en la descripción candidata que también se encuentran en la descripción de referencia, y ω_t es el número de unigramas en la descripción candidata. El recall de Unigrama R se calcula como:

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

Donde m es como arriba, y ω_r es el número de unigramas en la descripción de referencia. La precisión y el recall se combinan utilizando la media armónica de la siguiente manera, con un recall ponderado 9 veces más que la precisión:

$$Fmean = \frac{10\ P\ R}{R + 9\ P}$$

Hasta este momento las medidas descritas únicamente dan cuenta el respecto de palabras individuales, pero no respecto a segmentos más grandes que aparecen tanto en la descripción de referencia como en la descripción candidata. Para ello, se utilizan coincidencias de n-gramas más largas para calcular una penalización p para los alineamientos. Cuantas más asignaciones haya que no sean adyacentes entre la de referencia y la candidata, mayor será la penalización. Para calcular esta penalización, los unigramas se agrupan en la menor cantidad posible de fragmentos, donde un fragmento se define como un conjunto de unigramas que son adyacentes en la hipótesis y en la referencia. Cuanto más largas sean las asignaciones adyacentes entre el candidato y la referencia, menos fragmentos habrá. Una traducción idéntica a la referencia dará solo un fragmento. La penalización p se calcula de la siguiente forma:

$$p = 0.5\left(\frac{c}{\mathcal{U}_m}\right)$$

Donde c es el número de trozos y $\mathcal{U}_m$ es el número de unigramas que se han mapeado. La puntuación M final de un segmento se calcula tal y como se desprende a continuación. Teniendo en cuenta que la penalización introducida tiene el efecto de reducir la $Fmean$ hasta en un 50% si no hay bigramas o coincidencias más largas.

$$M = Fmean\ (1 - p)$$

En resumen, en el caso de METEOR, a diferencia de BLEU, se realiza la media armónica de la precisión y el recall por unigrama (el recall tiene mayor peso que la precisión). La ventaja respecto a BLEU es no requiere coincidencias exactas usa sinónimos. Para ello utiliza Wordnet. Está diseñado para resolver algunos de los problemas de BLEU. Cuanto mayor sea la puntuación METEOR, mejor será el rendimiento del sistema. Para calcular una puntuación de más de todo un conjunto de datos, los valores de P, R y p se van almacenando, y finalmente se combinan utilizando la misma fórmula

- **CIDEr (Consensus-based Image Description Evaluation)** [12]**:** es una métrica especialmente diseñada para evaluar descripciones de imágenes. Todas las palabras de las descripciones (tanto candidatas como referencias) se transforman a su respectivo lema o raíz, para ampliar la búsqueda de n-gramas a no solo coincidencias exactas. Además, CIDEr se basa en la similitud coseno. Mide la coherencia de la anotación de imágenes realizando un cálculo del peso de la Frecuencia de Términos-Frecuencia Inversa de Documentos (TF-IDF) para cada n-grama. Este indicador trata cada descripción como un "documento", la representa en forma de vector TF-IDF y, a continuación, calcula la similitud del coseno de la descripción de referencia con la descripción generada por el modelo como puntuación. En otras palabras, es el modelo de espacio vectorial. Este indicador compensa una de las desventajas de BLEU de que todas las palabras coincidentes se trataban igual, pero en realidad, cuando algunas de las palabras deberían tener mayor importancia. De nuevo, cuanto mayor sea la puntuación CIDEr, mejor será el rendimiento.

Existen otras muchas métricas como: SPICE [13], mRank, R@k, PPLX, Average Precision (AP), IoU… En este trabajo se ha optado por utilizar BLEU, METEOR y ROUGE-L, principalmente por motivos de implementación de software, ya que muchas de las métricas mencionadas están disponibles como parte del servidor de evaluación de MSCOCO para permitir la evaluación sistemática y comparativa con el formato de datos que utiliza Coco. En cambio, BLEU, y METEOR están disponibles en la librería nltk y ROUGE-L en la librería py-rouge.

2.4. Resultados Flickr-30k

Por último, finalizamos esta sección de revisión del estado del arte con una comparativa de resultados utilizando el conjunto de datos Flickr30K, y las métricas BLEU-N (**B1**) (**B2**) (**B3**) y (**B4**) METEOR (**M**), y CIDEr (**C**) para la evaluación. Ésta no es una revisión exhaustiva, pero se han incluido los trabajos más citados que incluyen resultados comparables.

Method	B1	B2	B3	B4	M	C
Kiros et al. (2014b)	60	38	25.4	17.1	16.9	
Chen and Zitnick (2015)				12.6	16.4	
Donahue et al. (2015)	58.7	39.1	25.1	16.5		
Jia et al. (2015)	64.6	44.6	30.5	20.6	17.9	
Karpathy and Fei-Fei (2015)	57.3	36.9	24	15.7		
Mao and Yuille (2015)	60	41	28	19		
Vinyals et al. (2015)	66.3	42.3	27.7			
Xu et al. (2015) Soft	66.7	43.4	28.8	19.1	18.49	
Xu et al. (2015) Hard	66.9	43.9	29.6	19	18.5	
Oruganti et al. (2016)	58.9	40	26.6	17.7	17.8	
Wu et al. (2016)	73	55	40	28		
You et al. (2016)	64.7	46	32.4	23	18.9	
Chen et al. (2017a)	66.2	46.8	32.5	22.3	19.5	44.7
Gan et al. (2017b)	74.7	55.2	40.3	28.8		
Gu et al. (2017)	73.8	56.3	41.9	30.7	20.6	
Fu et al. (2017)	64.9	46.2	32.4	22.4	19.4	
Lu et al. (2017)	67.7	49.4	35.4	25.1	20.4	53.1
Li and Chen (2018)	**75.5**	**57.1**	**42.9**	**31.7**	**22.9**	**71.5**
Wang and Chan (2018)	57.7	40.1	27.6	19	18.4	35.2
Wang and Chan (2018) Hier	60.7	42.5	29.2	19.9	19.1	39.5

Tabla 3: Comparativa de resultados obtenidos en trabajos estado del arte con el conjunto de datos Flickr-30K

La gran mayoría de los resultados mostrados en la tabla 3 se han obtenido utilizando MSCOCO en entrenamiento y Fickr-30K para evaluar, o como complemento en el entrenamiento (Data Augmentation), por lo que seguramente los resultados serán significativamente superiores a los que se pueden obtener entrenando únicamente con Flickr-30k. El trabajo del cual se han destacado los resultados: Li and Chen (2018), profundiza en el Aprendizaje por Refuerzo mencionado anteriormente.

3. Redes neuronales para el tratamiento de secuencias: la arquitectura Codificador-Decodificador.

Recogiendo algunas de las ideas ya presentadas, la tarea de descripción automática de imágenes es un problema de Machine Learning en el que se debe generar una descripción textual para una fotografía determinada. Anteriormente, se mencionaban varios enfoques para abordar este problema, este trabajo se centra en los métodos basados en modelos del lenguaje, concretamente en las arquitecturas Codificador-Decodificador, por lo que se entiende el aprendizaje del modelo como un problema de optimización donde se trata de maximizar la probabilidad de que la descripción dada en la salida del modelo sea correcta. A continuación, se deprende la función objetivo, donde θ son los parámetros del modelo, I una imagen dada y S la descripción ideal:

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

En este enfoque se requiere la combinación de 2 métodos: por un lado, la Visión por Computador para comprender el contenido de la imagen y, por otro lado, un modelo de lenguaje del campo del Procesamiento del Lenguaje Natural para convertir la comprensión de la imagen en palabras en el orden correcto. Últimamente, las técnicas de Deep Learning han logrado resultados de estado del arte en este problema. Una red convolucional, actúa como codificador efectuando la extracción de características, siendo las características de la última capa totalmente conectada o capa convolucional extraídas como características de la imagen. Esta información llega al decodificador que es una red neuronal recurrente, que actúa como modelo del lenguaje para la generación de la descripción de la imagen. Debido a la complejidad del entrenamiento de la RNN [44], y a que hay un problema general de descenso de gradiente, que, aunque puede ser ligeramente compensado por la regularización [45], la RNN sigue teniendo un defecto fatal que es que sólo puede recordar el contenido de la unidad de tiempo limitada anterior. Así la estructura de una RNN sencilla es la siguiente:

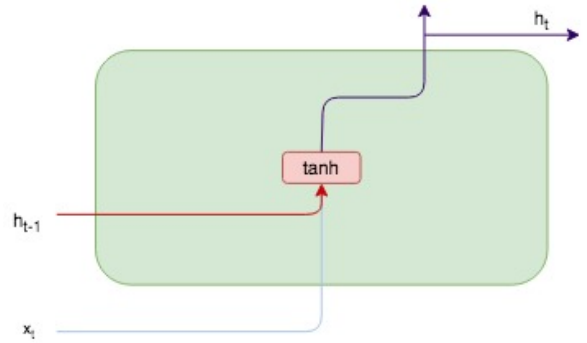

Ilustración 5: Esquema de funcionamiento de una RNN sencilla

Se puede observar una celda con dos entradas y dos salidas. $\mathbf{x_t}$ es la entrada con nueva información para la red, el siguiente elemento de la secuencia a tratar. $\mathbf{h_{t-1}}$ es la salida de esta celda para el instante anterior a t. Ambos flujos se concatenan. Dentro de la celda tenemos una capa completamente conectada con una tangente hiperbólica como función de salida. Ésta diverge en dos, un camino es la salida en este instante t y otro el estado

oculto que recibirá la red recurrente como retroalimentación en el instante t+1. La función de salida en la RNN es:

$$h(t) = \sigma(U_h\, x_t + V_h\, h_{t-1} + b_h)$$

De esta forma, se consigue conectar la información a lo largo del tiempo. Pero en la práctica esto no se cumple del todo al menos, ya que mantener información dependiente a lo largo de varios instantes t se hace complicado al desvanecerse rápidamente la información anterior en cada t. Actualmente, se utilizan otro tipo de RNN como es la Long-Short Term Memory (LSTM) [37].

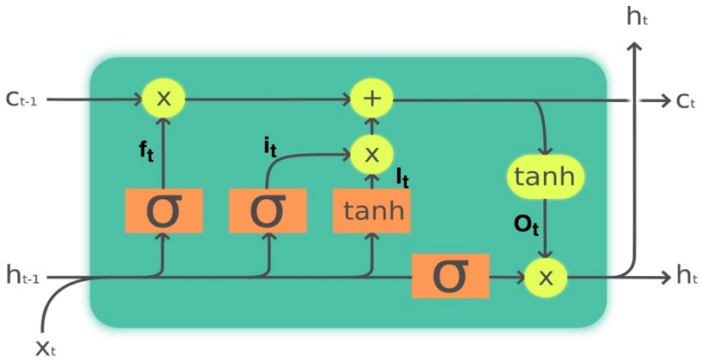

Ilustración 6: Esquema de funcionamiento de una LSTM

El principal cambio que introduce la LSTM respecto a la RNN sencilla descrita anteriormente, es que el flujo $\mathbf{C_{t-1}}$ a $\mathbf{C_t}$, que permite salvaguardar información entre instantes t sin perder información antigua si así se desea. Esto es posible debido a que hay un flujo casi directo si no fuera por dos entradas de nuevos flujos denominados como puertas o "gates". La primera es la puerta f_t conocida como "forget gate", o puerta de olvido (**fg**). Es la encargada de regular que información del flujo del instante anterior entra en este instante. Una red neuronal con una función sigmoide al final ($\mathbf{f_t}$) regula $\mathbf{C_{t-1}}$, al estar ft en un intervalo [0,1], donde 0 indica ignorar totalmente $\mathbf{C_{t-1}}$ y 1 conservarlo. Mientras que los pesos de la red decidirá en base al input actual y el output del instante anterior el valor en dicho intervalo.

$$fg(C_{t-1}) = C_{t-1}\, f_t$$

$$f_t = \sigma(W_t\, h_{t-1} + V_t\, x_t + b_t)$$

El siguiente punto de modificación añade información nueva al vector. Este nuevo flujo es igual que el que teníamos en la RNN clásica, una capa completamente conectada con una tangente hiperbólica en la última capa ($\mathbf{l_t}$). La "input gate", o puerta de entrada (**ig**) es la encargada de regular la salida ($\mathbf{i_t}$) de esta red, determinando que nueva información se añade al flujo directo. Esta nueva puerta, al igual que la puerta de olvido, es una suma ponderada con una función sigmoide en la última capa y $\mathbf{h_{t-1}}$ concatenada a $\mathbf{x_t}$ para luego multiplicar la salida con el vector de salida $\mathbf{l_t}$:

$$ig(C_{t-1}) = i_t\, l_t$$

$$i_t = \sigma(K_t\, h_{t-1} + T_t\, x_t + c_t)$$

$$l_t = tanh(J_t * h_{t-1} + U_t\, x_t + d_t)$$

De esta manera, el flujo directo al final del instante t queda de la siguiente forma:

$$C_t = fg(C_{t-1}) + ig(C_{t-1})$$

Hay que tener en cuenta que la salida del flujo directo no puede ser confundida con la salida de la celda en el instante t. Por un lado, el flujo conectará directamente con la próxima celda en el instante t+1, mientras que la salida de la celda primero pasará por una función tangente hiperbólica que situará los valores del vector en el intervalo [-1,1]. Tras esto una tercera puerta conocida como "output gate", o puerta de salida (**og**) decidirá que parte de la información saldrá de esta celda, de forma idéntica a las otras dos gates. Esta salida irá al output de la celda por un lado y por otro se comunicará con la siguiente celda.

$$h_t = og(t)\, O_t$$

$$og(t) = \sigma(P_t\, h_{t-1} + E_t\, x_t + e_t)$$

$$O_t = tanh(C_t)$$

Tal y como se ha explicado la LSTM [46] es una arquitectura especial de RNN que puede resolver problemas como la desaparición del gradiente, y tiene memoria a largo plazo. En los últimos años, la red LSTM ha tenido un buen rendimiento en el tratamiento del contexto relacionado con el vídeo [47] [48] [49]. Al igual que el contexto de vídeo, la estructura del modelo LSTM de la ilustración 5 se utiliza generalmente en la etapa de decodificación.

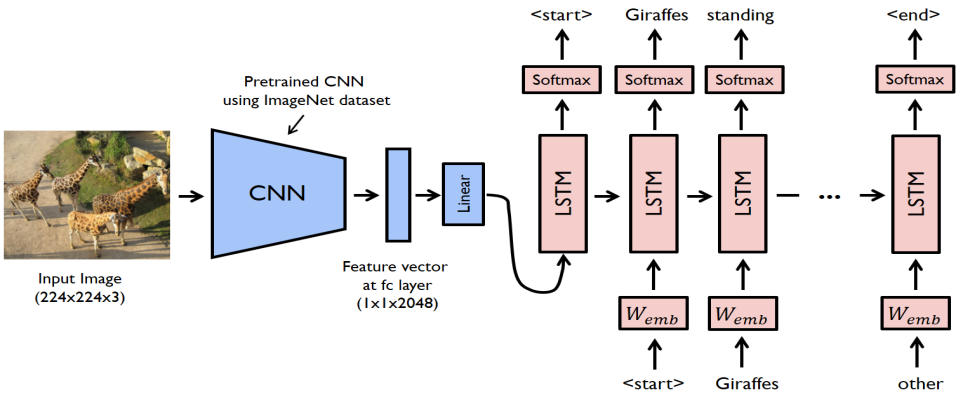

Ilustración 7: Arquitectura Codificador-Decodificador para la tarea de descripción de imágenes [50]

En la ilustración 7 se observa una de las primeras aproximaciones con Deep Learning a este problema. Se muestra la recurrencia temporal de la RNN, que en este caso es una LSTM, para generar una descripción, seleccionando la palabra más probable dentro del vocabulario, desde la secuencia de inicio <start>, hasta que la secuencia de fin <end> es la palabra más probable. En esta propuesta el vector de características obtenido por la red convolucional únicamente sirve como inicialización de la LSTM, es decir, no es parte de la entrada recurrente que alimenta la LSTM en cada paso temporal para generar cada una de las palabras de la descripción.

Para comprender un poco mejor esta aproximación y poder entrar más al detalle es necesario conocer el término de "Embeddings" que aparecen en la ilustración como "Wemb". A la hora de codificar las palabras para que un sistema Image Captioning pueda procesarlo es usual recurrir al One-Hot encoding, este sistema se basa en que cada palabra viene representada por un vector binario, donde todos los valores son 0's menos cierta posición, única para cada palabra, con valor 1. Por lo tanto, para representar una colección de n palabras será necesario utilizar vectores de longitud n. El problema de utilizar este tipo de codificación, es que la dimensionalidad de esta representación escala mal, encontrándose la mayoría de posiciones infrautilizadas. Además de que dichos vectores tienen escasa expresividad en cuanto a su significado. Por ello, en la actualidad se suele utilizar embeddings para la codificación de palabras.

Un embedding, es una representación de la palabra que ha sido mapeada a un espacio vectorial donde palabras similares se sitúan cerca, mientras que palabras diferentes estarán alejadas. Es decir, se trata de realizar una codificación, que más allá de hacerlas procesables por sistemas computacionales, ofrezca información acerca de la semántica de las palabras que representan. La formación de embeddings requiere de un proceso de entrenamiento previo a partir de un corpus que contenga todas las palabras a mapear en su contexto natural. Esto es esencial ya que los embeddings se basan en el contexto que rodea a cada palabra. Uno de los algoritmos más comunes a la hora de entrenar los embeddings es el conocido como Word2Vec, el cual fue presentado en Mikolov, et al. [51] .Este algoritmo se basa en utilizar una red neuronal completamente conectada de una sola capa para que realice el mapeo desde una palabra codificada en formato One-Hot al embedding. Tras entrenar a la red, el embedding será el resultado obtenido en la salida de la red tras haber introducido la palabra en ella o lo que es lo mismo, el embeddings es el contexto aprendido por el modelo para esa palabra.

Ahora ya estamos en mejor disposición para entrar al detalle del proceso seguido en la ilustración 7. El instante t = −1 es el momento en el que la imagen se procesa a través de la CNN para entrar como input de la LSTM (x−1). En este instante la LSTM no genera ningún output más que los flujos recurrentes propios de la LSTM. Los inputs de la LSTM son Embeddings (Wemb St), siendo St la palabra en el instante t. Entonces, lo que podemos tomar x−1 como el embedding de la imagen en el mismo espacio que los embeddings de las palabras. El output de la LSTM (las palabras del vocabulario en cada instante – pt) es la distribución probabilística a lo largo de todo el vocabulario para la posición t en la descripción generada. Por lo que a la hora de realizar el entrenamiento la función loss es la suma de la probabilidad logarítmica negativa de la palabra correcta en cada paso de la generación:

$$L(I,S) = -\sum_{t-1}^{N} \log p_t(S_t)$$

En el momento de entrenar el modelo el input de la LSTM es el embedding de la palabra correcta en la posición anterior, mientras que al evaluar un modelo el input será el embedding de la palabra más probable según el criterio de la LSTM en el t − 1. El input en t = 0 debe ser una palabra estándar para todas las descripciones ya que no hay ningún criterio para poder decidir dicho valor en cada descripción, por ello se suele recibir como S0 un marcador de inicio de la frase <start>.

No obstante, esta aproximación se ha visto claramente mejorada por otras propuestas que sugieren innovaciones como: utilizar la información del vector de características en cada una de las recurrencias durante la generación de una descripción, o añadir un modelo de atención capaz de seleccionar la información más relevante de la representación de la imagen y las palabras anteriores de la descripción, etc. Lo más asombroso de estos métodos es que se puede definir un único sistema o modelo END-TO-END para predecir una descripción dada una imagen, en lugar de requerir una preparación de datos sofisticada o una serie de modelos diseñados para subtareas específicas. En este caso se define un aprendizaje basado en un modelo de fusión "Merge Model" descrito por Marc Tanti, et al. en sus artículos de 2017 [52] [53]. Para comprender mejor esta arquitectura puede verse la siguiente publicación:

"Caption Generation with the Inject and Merge Architectures for the Encoder-Decoder Model" [54].

Los autores proporcionan un esquema del modelo de fusión que se reproduce en la implementación de los modelos del trabajo:

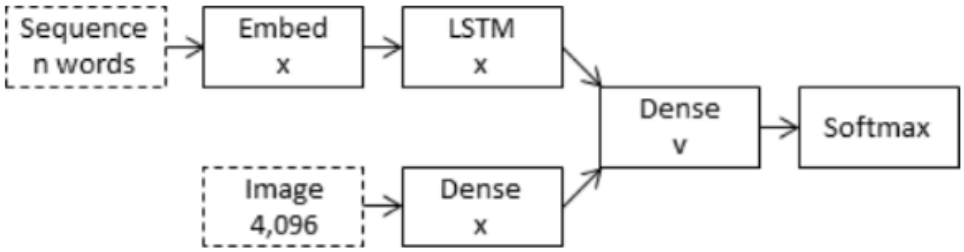

Ilustración 8: Arquitectura Merge (de fusión) para el modelo codificador-decodificador [54]

El problema de este método es que, cuando el modelo implementado intenta generar la siguiente palabra de la descripción, ésta suele describir sólo una parte de la imagen (ilustración 3). Es incapaz de captar la esencia de toda la imagen de entrada. Utilizar la representación completa de la imagen para condicionar la generación de cada palabra no puede producir de forma eficiente diferentes palabras para diferentes partes de la imagen. Por este motivo es útil la utilización de un mecanismo de Atención.

Con un mecanismo de atención, la imagen se divide primero en n partes, y calculamos con una Red Neural Convolucional (CNN) las representaciones de cada parte h1, ..., hn. Cuando la RNN está generando una nueva palabra, el mecanismo de atención se centra en la parte relevante de la imagen, por lo que el decodificador sólo utiliza partes específicas de la imagen.

Con una nueva capa de modelo de atención. Si hemos predicho i palabras, el estado oculto de la LSTM es hi. Seleccionamos la parte "relevante" de la imagen utilizando hi como contexto. A continuación, la salida del modelo de atención zi, que es la representación de la imagen filtrada de forma que sólo quedan las partes relevantes de la imagen, se utiliza como entrada para el LSTM. A continuación, la LSTM predice una nueva palabra devolviendo un nuevo estado oculto hi+1.

3.1. Mecanismo de atención

El mecanismo de atención, derivado del estudio de la visión humana, es una habilidad cognitiva compleja que tenemos los seres humanos en la neurología cognitiva. Cuando las personas reciben información, pueden ignorar conscientemente parte de la información principal mientras ignoran otra información secundaria. Esta capacidad de autoselección se llama atención. Este mecanismo se propuso por primera vez para ser aplicado a la clasificación de imágenes en el campo de las imágenes visuales utilizando el mecanismo de atención en el modelo RNN [55].

En el procesamiento del lenguaje natural, cuando las personas leen textos largos, la atención humana se centra en palabras clave, eventos o entidades. Una gran cantidad de experimentos han demostrado que el mecanismo de atención se aplica en el procesamiento de texto, por ejemplo, la traducción automática [56] [57], generación de resúmenes [58] [59], comprensión del texto [60] [61] [62] [63], clasificación del texto [64] [65] [66], subtítulos visuales [67] [68] y otras cuestiones, los resultados obtenidos son notables, y a continuación se mencionan la aplicación de diferentes métodos de mecanismos de atención en el marco básico de descripción de imágenes, de modo que se mejore la calidad del sistema. Como se anticipaba previamente, en los modelos de redes neuronales, la introducción del mecanismo de atención consiste en dotar a la red neuronal de la capacidad de concentrarse en su subconjunto de entradas (o características), para seleccionar aquellas más relevantes en cada instante para la generación de cada palabra de la descripción. Basándose en las ventajas del mecanismo de atención mencionadas anteriormente, se muestran los diversos mecanismos de atención:

Nombre	Método	Descripción
Soft attention [69]	Dar una probabilidad de acuerdo con el vector de contexto para cualquier palabra en la oración de entrada al buscar la distribución de probabilidad de atención	Habilitación derivada de parametrización Definitivamente
Hard attention [69]	Concéntrese solo en una ubicación elegida al azar utilizando el muestreo de Monte Carlo para estimar el gradiente	Aleatoriamente Sobre la base de la probabilidad Simple
Multihead attention [70]	Proyectar linealmente múltiples piezas de información seleccionadas de la entrada en paralelo usando múltiples claves, valores y consultas	Proyección lineal Paralelo Enfoque en la información de diferentes subespacios de representación en diferentes ubicaciones Cabeza de atención múltiple
Scaled dot-product attention [70]	Ejecute una única función de atención utilizando claves, valores y matrices de consulta.	Alta velocidad Ahorre espacio
Global attention [71]	Teniendo en cuenta el estado de la capa oculta de todos los codificadores, la distribución del peso de la atención se obtiene comparando el estado de la capa oculta del decodificador actual con el estado de cada capa oculta del codificador.	Completo Consume mucho tiempo Gran cantidad de cálculos
Local attention [71]	Primero encuentre la zona de la imagen interesante, luego calcule el peso de atención en las ventanas izquierda y derecha de su ubicación, y finalmente pondere el vector de contexto	Reducir el costo de los cálculos
Adaptive attention [72]	Defina un nuevo vector de contexto adaptativo que se modele como una mezcla de las características de la imagen atendidas espacialmente y el vector centinela visual. Compensa la cantidad de información nueva que la red está considerando a partir de la imagen con lo que ya conoce en la memoria del decodificador.	Resolver cuándo y dónde llamar la atención para extraer información significativa para las palabras de la secuencia.
Semantic attention [73]	Seleccione conceptos semánticos e incorpórelos en el estado oculto y la salida del LSTM	Combinación opcional De arriba a abajo De abajo hacia arriba

Tabla 4: Comparación de algunos de los métodos más conocidos de modelado del mecanismo de atención

En este trabajo se ha optado por la implementación de un mecanismo de atención local, para tratar de mejorar los resultados del enfoque clásico sin mecanismo de atención.

Atención local o Bahdanau

En Bahdanau o atención local [71], la atención se centra sólo en unas pocas posiciones de la fuente. Como la atención global se centra en todas las posiciones de la fuente para obtener cada una de las palabras objetivo, es computacionalmente muy cara. Para superar esta deficiencia, la atención local opta por centrarse sólo en un pequeño subconjunto de los estados ocultos del codificador por palabra objetivo. Primero, encuentra una posición de alineación y luego calcula el peso de la atención en las ventanas izquierda y derecha donde se encuentra su posición y finalmente pondera el vector de contexto. En realidad, se trata de un compromiso mixto entre soft y hard. La principal ventaja de la atención local es reducir el coste del cálculo del mecanismo de atención. En el cálculo, la atención local no consiste en considerar todas las palabras del lado de la lengua de origen, sino en predecir la posición del extremo de la lengua de origen que se va a alinear en la decodificación actual según una función de predicción y, a continuación, navegar por la ventana de contexto, considerando sólo las palabras que se encuentran dentro de la ventana.

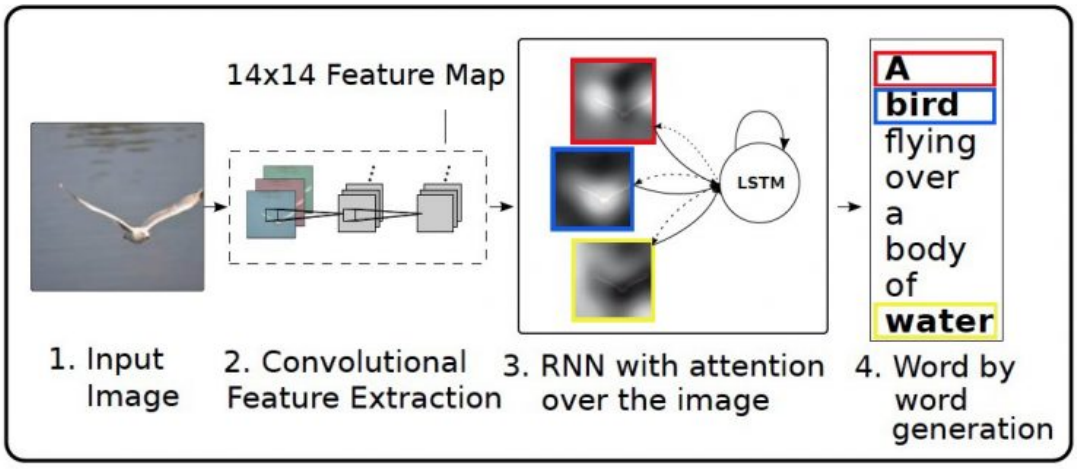

Ilustración 9: Esquema básico de funcionamiento de un sistema de descripción de Imágenes con arquitectura Codificador-Decodificador y mecanismo de Atención local [74]

Todos los estados ocultos del codificador y el descodificador se utilizan para generar el vector de contexto. El mecanismo de atención alinea las secuencias de entrada y salida, con una puntuación de alineación parametrizada por una red de retroalimentación. Ayuda a prestar atención a la información más relevante en la secuencia fuente. El modelo predice una palabra objetivo basándose en los vectores de contexto asociados con la posición de origen y las palabras objetivo generadas previamente. En este caso, varias imágenes equivalen a varias oraciones en el idioma de origen en la traducción.

4. Marco Experimental

Las diferentes implementaciones de este trabajo se han realizado bajo la plataforma de software Python. Por un lado, se hace uso de la biblioteca de código abierto TensorFlow, orientada al entrenamiento de modelos de Aprendizaje automático profundo, y el framework de alto nivel para el aprendizaje, Keras. Por otra parte, se ha empleado el toolkit de lenguaje natural NLTK, que es un conjunto de bibliotecas de software libre para el lenguaje de programación Python. También, otras librerías auxiliares necesarias como: py-rouge, numpy, matplotlib, os, glob, etc.

4.1. Procesado de datos

Conjunto de datos

Para el desarrollo de este trabajo se ha utilizado el conjunto de datos "**Flickr-30k"**, que tal y como se adelantaba en el capitulo de estado del arte, consta de 31.782 imágenes con 5 descripciones por imagen en inglés proporcionadas por anotadores humanos. Al asociar cada imagen con varias oraciones producidas de forma independiente, el conjunto de datos captura parte de la variedad lingüística que se puede utilizar para describir la misma imagen. Dicho conjunto, consta de un directorio con las 31.782 imágenes, por un lado y por otro, un fichero de texto "Flickr30k.token.txt" que contiene tanto el nombre o identificador de las imágenes, como los subtítulos asociados. Véase un ejemplo de esto:

1000092795.jpg#0	Two young guys with shaggy hair look at their hands while hanging out in the yard.
1000092795.jpg#1	Two young, White males are outside near many bushes.
1000092795.jpg#2	Two men in green shirts are standing in a yard.
1000092795.jpg#3	A man in a blue shirt standing in a garden.
1000092795.jpg#4	Two friends enjoy time spent together.

El conjunto de datos está disponible de forma gratuita. Se debe completar un formulario de solicitud y los enlaces al conjunto de datos se le enviarán por correo electrónico. No vienen vinculados en este trabajo debido a que, la dirección de correo electrónico que proporciona los datos, solicita expresamente: "Por favor, no redistribuya el conjunto de datos ".

Tras investigar detenidamente sobre como establecer la partición del dataset para poder establecer una comparativa de resultados consecuente, la única publicación encontrada que mencionaba algo al respecto es el papel de V. Mullachery, V. Motwani (2018) [75] que propone una partición algo inexacta. Plantea dividir las 30k muestras del conjunto en: 28k para Entrenamiento y 2k para Pruebas. Sin embargo, hay 31.782 datos, por lo que se ha optado por utilizar aproximadamente una partición típica de (80-10-10) o (90-10).

	Total	**Entrenamiento**	**Validación**	**Prueba**
imágenes	31.782	25.426	3.178	3.178
Descripciones asociadas	158.910	127.130	15.890	15.890

Tabla 5: Resumen de la partición del conjunto de datos Flickr-30k (imágenes)

Esta partición se genera manualmente generando 1 fichero de texto que contiene 31.782 líneas, cada una con el identificador de una de las imágenes diferentes del conjunto de datos. Se realiza un barajado y la partición de este fichero en 3: train_set.txt (25.426 ids), dev_set.txt (3.178 ids), y test_set.txt (3.178 ids).

Codificación de las imágenes

Para la tarea de extracción de características de la imagen, se ha optado por utilizar una DenseNet121 pre-entrenada con imageNet. Se elimina la última capa de clasificación, y se utiliza el vector de 1024 características obtenido en la penúltima capa como entrada de la RNN:

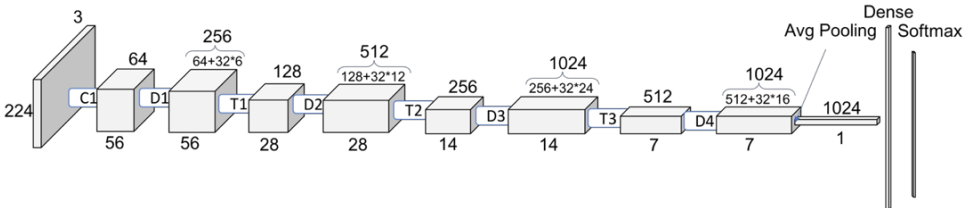

Ilustración 10: Esquema simplificado de la arquitectura interna de la topología de CNN DenseNet-121 [76]

Este tipo de redes se denominan dense porque dentro de un mismo bloque se concatenan las salidas de todas las capas convolucionales anteriores, a la entrada de todas las siguientes, tal y como se desprende a continuación:

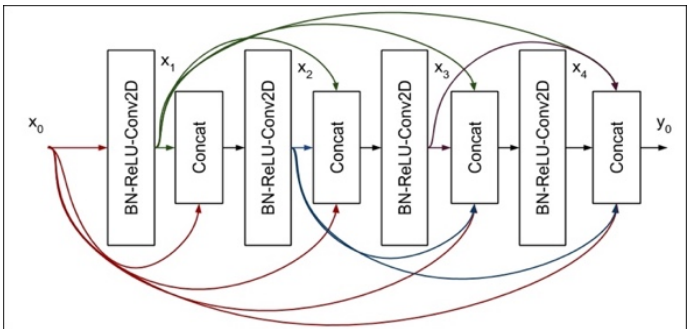

Ilustración 11: Arquitectura interna de un bloque Dense de 4 capas en una CNN DenseNet [77]

Se ha optado por esta CNN debido a que es una red suficientemente potente (mas potentes que las ResNet), y generalmente obtiene incluso mejores resultados que otras topologías más potentes, como InceptionV3, además de ser más rápida de entrenar, en el caso de que tengamos que hacer Fine-tuning de estos pesos. No obstante, en este trabajo solo se va a utilizar para inferir y codificar las imágenes.

Hasta este momento, no se ha mencionado que las imágenes del dataset tienen tamaños variables, por lo que habrá que hacer un resizing de todas ellas para adaptarlas al tamaño de entrada de la red: 224x224x3. Además, habrá que tener en cuenta algunas otras consideraciones necesarias. Todo esto viene definido en una función propia llamada **preprocess_img().** También se implementa otra función llamada **encode()** que recibe la ruta de la imagen a codificar y devuelve la imagen codificada (vector de 1024-d).

Mediante la librería glob se genera un listado con las rutas de cada una de las imágenes del dataset. Y, con este listado de rutas y los ficheros generados a mano con las ids de las imágenes divididas en entrenamiento, validación, y prueba (train_set.txt, dev_set.txt, y test_set.txt) se procede a realizar la codificación de las imágenes en 3 conjuntos, concretamente 3 numpy.arrays (train_features, dev_features y test_features) que contienen los vectores de 1024-d correspondientes a las imágenes codificadas. De esta forma, ahorramos tiempo en el entrenamiento del Decoder, y le proporcionamos como entrada de la red las imágenes ya codificadas, como más adelante veremos.

Preprocesado de Texto

El conjunto de datos contiene múltiples descripciones para cada imagen, concretamente 5 por cada una. Este texto de las descripciones requiere una limpieza mínima. Teniendo en cuenta que no se trata de una tarea de tweets ni nada similar, no serán necesarios tokenizadores muy sofisticados, ni una limpieza de texto demasiado exhaustiva.

1) En primer lugar, se carga el archivo que contiene todas las descripciones con una función denominada **load_file().** La función recibe la ruta del fichero de texto con las descripciones, y devuelve una variable de tipo texto leyendo cada una de las líneas del fichero proporcionado. Tal y como se ha detallado anteriormente, cada imagen tiene un identificador único (id). Este identificador se utiliza como nomenclatura de la imagen y de referencia en el archivo de texto de descripciones.

2) A continuación, se recorre la lista de descripciones de las fotografías. Para ello, se define otra función denominada **load_description()** que, toma el texto cargado, y devuelve un diccionario Python que tendrá como claves los identificadores de las imágenes y como valores la lista de descripciones asociadas a esa id. Cada identificador de imagen se asigna a una lista de una o más descripciones textuales (Generalmente 5: #0, #1, #2, #3, #4):

   Formato diccionario Python de salida:

   {'img_id_1': (descripcion_0, descripcion_1, descripcion_2, descripcion_3, descripcion_4) ...}

3) El siguiente paso es generar 3 listas que contengan todas las descripciones, añadiendo los tokens de inicio y fin de secuencia: <start> y <end> al principio y final de cada una de las descripciones. Nuevamente, se utilizan los ficheros de partición del corpus, y el diccionario obtenido en el paso anterior para generar los 3 listados: train_descriptions, dev_descriptions, y test_descriptions.

4) Se prosigue con el tokenizado y conversión a secuencia. Para ello, se han empleado las herramientas de keras.preprocessing. En primer lugar, se utiliza el tokenizador de keras para tokenizar las descripciones y generar el vocabulario. Seguidamente, se realiza la conversión Text-To-Sequence donde se utiliza padding posterior para adaptar todas las secuencias correspondientes a las descripciones, a un tamaño de descripción determinado. El padding consiste en añadir 0 a los vectores generados, para que todas las secuencias generadas tengan la misma dimensionalidad, pese a que las descripciones originales tuvieran diferente numero de palabras. Para ajustar este parámetro de tamaño de descripción se ha realizado un análisis de las diferentes longitudes de descripciones del dataset:

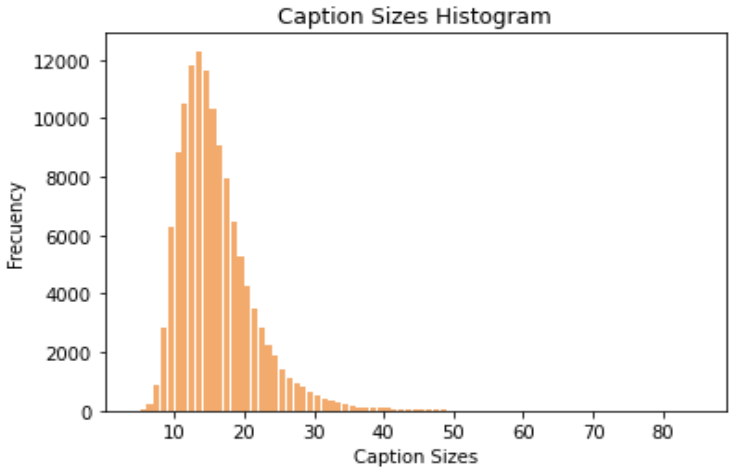

Ilustración 12: Histográma de longitud de descripciones

La longitud de una descripción máxima es de 85 palabras, sin embargo, se puede apreciar que la gran mayoría de descripciones no pasan de las 25, 30 palabras. Teniendo en cuenta que uno de los principales problemas de las RNN es que no son capaces de aprovechar la paralelización de cálculos con GPU, decantarse por un tamaño de secuencia de 85-D podría prolongar en exceso el entrenamiento. Por este motivo se ha decidido utilizar un tamaño de secuencia de 25.

5) Posteriormente, se ha cargando un fichero de texto que contiene un conjunto de Word Embeddings de 200 dimensiones pre-entrenados para una tarea de Wikipedia y Gigaword5: "glove.6b.200d.txt (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors, 822 MB download)" [78]. Este fichero se ha obtenido de la web del grupo de procesamiento de lenguaje natural de la universidad de Stanford (GloVe: Global Vectors for Word Representation) [79]. Con estos Embeddings y el tokenizador se genera una matriz de Embeddings de las palabras del vocabulario generado. Esta matriz se carga más adelante como pesos de la capa de Embeddings de nuestros modelos neuronales.

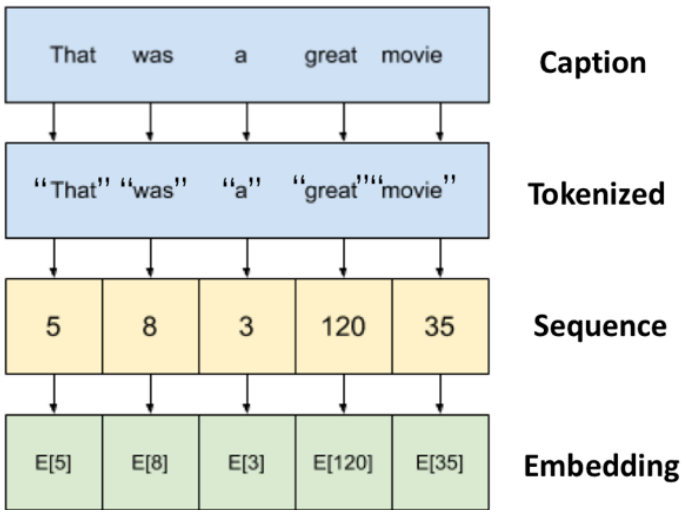

Ilustración 13: Esquema resumen del proceso de conversión desde la descripción hasta una secuencia de Embeddings

4.1. Topologías propuestas

Inicialmente, se definen 3 topologías a comparar sin modelo de atención. Las 3 siguen la misma filosofía: utilizar la información de la imagen codificada, en cada una de las recurrencias de la red recurrente. Se van a comparar los resultados obtenidos introduciendo esta información de 3 formas diferentes. Posteriormente, se introduce un mecanismo de atención local, realizando algunos cambios que se explican con más detalle más tarde.

Topologia 1

En la primera propuesta el modelo extractor de características de la imagen obtiene el vector de 1024 elementos. Estos son procesados por una capa Densa para producir una representación de 128 elementos de la imagen. Y este vector a su vez, se concatena a la salida de la LSTM en cada instante de tiempo.

Ambos modelos de entrada producen un vector de 128 elementos. Además, los dos utilizan la regularización y Dropout. Esto permite reducir el sobreajuste del conjunto de datos de entrenamiento, tal y como ya se ha explicado.

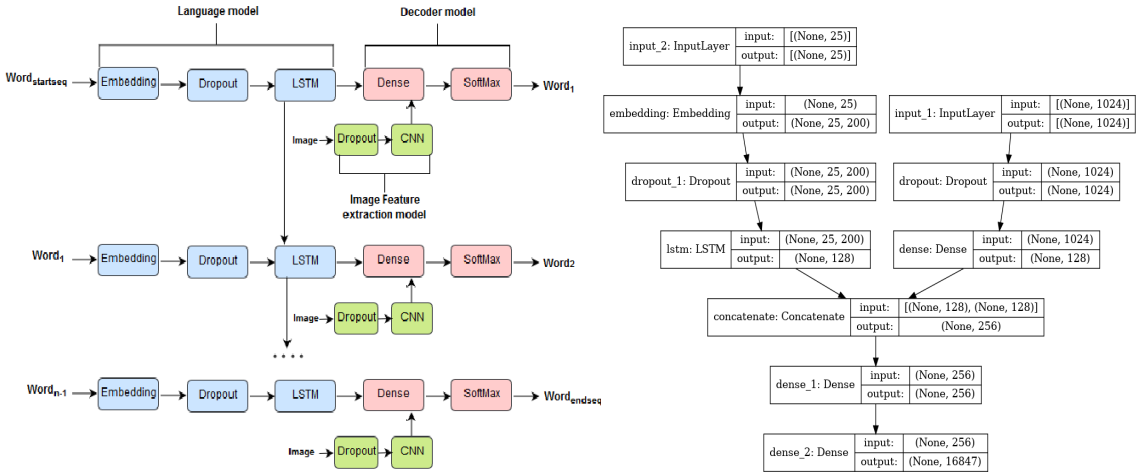

Ilustración 14: Esquema Arquitectura interna Topología 1 propuesta

El proceso seguido para la descripción de imágenes se divide en 3 partes:

- **Extractor de características de la imagen**: que como ya se ha explicado se realiza a través de una DenseNet121 con pesos pre-entrenados con imageNet.

- **Procesador de secuencias**: Se sigue el proceso descrito en la "Ilustración 13: Esquema resumen del proceso de conversión desde la descripción hasta una secuencia de Embeddings" tras cargar los pesos de la matriz de Word Embeddings generada en la capa de Embedding del modelo. De esta forma es posible manejar la entrada de texto, esta capa va seguida de la una capa de red neuronal recurrente Long Short-Term Memory (LSTM).

- **Decodificador**: Tanto el extractor de características como el procesador de secuencias generan un vector de longitud fija. Ambos vectores se fusionan y se procesan mediante una capa densa para hacer una predicción final

El modelo procesador de secuencias espera secuencias de entrada con una longitud predefinida (P palabras) que se introducen en una capa embedding. Esta capa utiliza una máscara para ignorar los valores rellenados (padding posterior). A esto le sigue una capa LSTM con N Neuronas. El Decodificador es el encargado de fusionar los vectores de ambos modelos de entrada mediante una operación de concatenación. Finalmente, hay una capa densa de salida final con función de activación Softmax, donde se realiza la predicción sobre todo el vocabulario de salida (186.847 palabras) para la siguiente palabra en la secuencia.

Topologia 2

Sigue la misma idea que el modelo anterior, pero en este caso la información de la imagen codificada se concatena antes de la LSTM. Para poder concatenar la salida de la capa de Embedding (Matriz 25 x 200) donde 25 es la longitud de secuencia y 200 la dimensión de los Word Embeddings, con el vector de características de la imagen, es necesario convertir este vector, a las mismas dimensiones. Para este fin se utiliza la capa repeat vector. Véase el proceso:

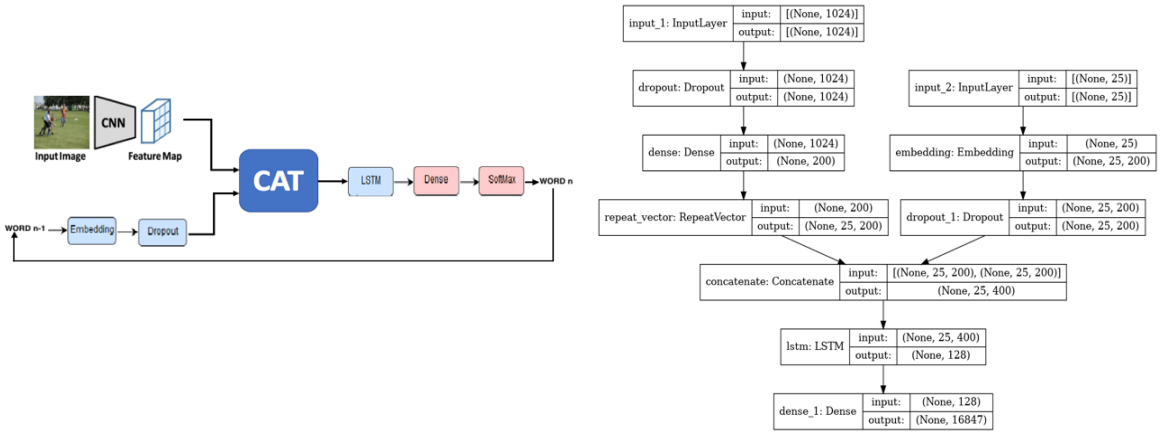

Ilustración 15: Esquema Arquitectura interna Topología 2 propuesta

Topologia 3

La última de las topologías propuestas consiste en combinar las dos aproximaciones anteriores, y concatenar la información de la imagen codificada antes y después de la LSTM.

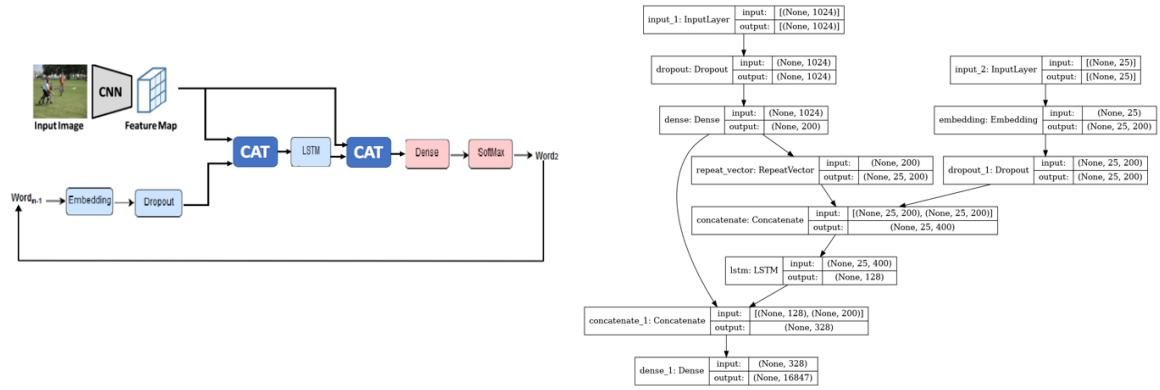

Ilustración 16: Esquema Arquitectura interna Topología 3 propuesta

Topología 4 (Mecanismo de atención local)

Continúa con la misma filosofía, pero introduciendo una serie de cambios que permitan la aplicación de un mecanismo de atención local. En lugar, de obtener un vector de características que represente la imagen, la imagen se divide primero en 7x7 = 49 trozos, y con la DenseNet121 se calcula el vector de características de cada uno de los trozos, siendo sus representaciones h1, ..., hn. Cuando la LSTM está generando una nueva palabra, el mecanismo de atención se centra en la parte relevante de la imagen, por lo que el decodificador sólo utiliza partes específicas de la imagen. Además de esta información, el modelo de atención recibe las palabras anteriores de la descripción y con todo ello genera un vector de contexto, similar al vector concatenado en las 3 topologías anteriores, pero con información más selectiva. A continuación, la LSTM predice una nueva palabra devolviendo un nuevo estado oculto hi+1.

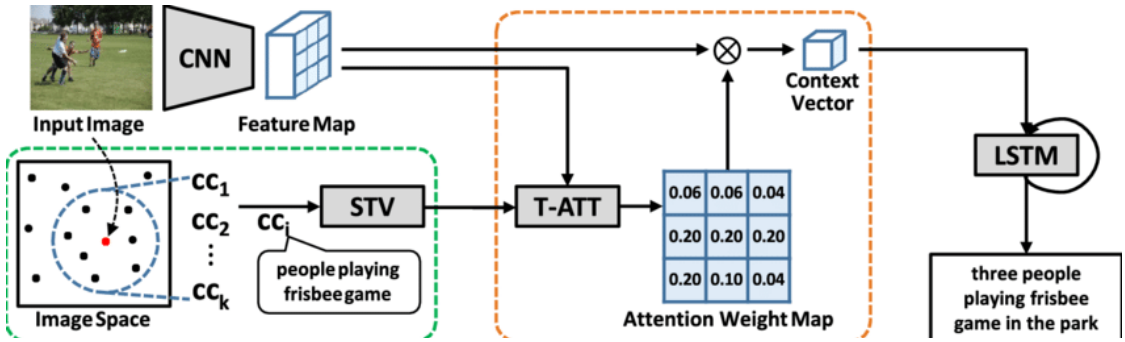

Ilustración 17: Topología 4: Ejemplo funcionamiento del modelo Codificador-Decodificador implementado con Mecanismo de atención local [80]

4.2. Detalles del Entrenamiento

Una vez finalizada la preparación de los datos: codificación de las imágenes, preprocesado y codificación del texto (descripciones), etc. se procede a preparar la fase de entrenamiento. Se ha realizado una experimentación previa, donde se ha modificando la topología de red inicial utilizando algunas técnicas para evitar, en la medida de lo posible, el sobreajuste (overfitting):

- **Regularization L1, L2:** La regularización consiste en aplicar al criterio de optimización empleado una penalización para evitar valores extremos de los pesos de la red, de esta forma se pretende obtener probabilidades altas con valores de los pesos moderados (Corrección del Overfitting). L1 y L2 son las más comunes, se obtienen calculando la norma en r del vector de pesos. Consiste en elevar a un termino r (en este caso 1 y 2) el valor absoluto de cada uno de los pesos y hago la raíz de r del sumatorio de todos estos términos. De esta forma L1 (r=1) será el sumatorio de los valores absolutos de los pesos, y L2 (r=2) la raíz cuadrada del sumatorio del cuadrado de los pesos. Estos términos (L1, L2) se multiplican por un factor de escala con valores del orden entre 10^{-2} y 10^{-5}. Esta regularización se aplica sobre las capas densas:

  model.add(Dense(1024, kernel_regularizer= regularizers.l1_l2(l1=1e-5, l2=1e-4))) [81]

- **Dropout** [82]: es otra de las posibles soluciones para conseguir una mejor generalización. Consiste en eliminar neuronas aleatoriamente, forzando a que las neuronas no eliminadas tengan la misma capacidad de representación que tendrían todas. Similar al Sparsity pero menos agresivo, en este caso las neuronas desactivadas rondan el 50%.

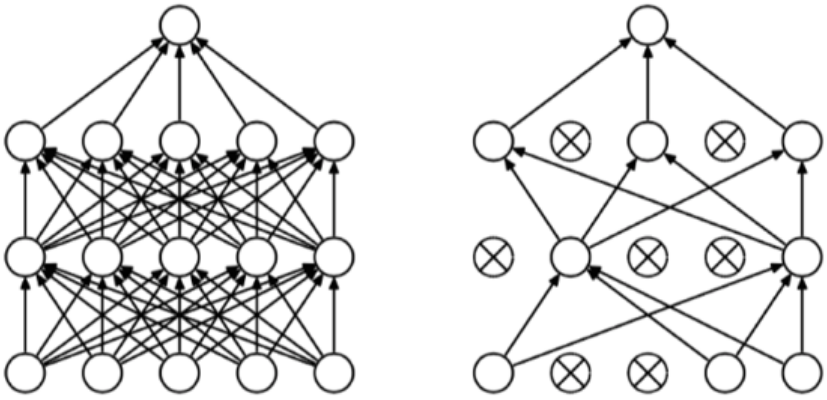

Ilustración 18: Ejemplo de funcionamiento del modelo neuronal con Dropout [82]

Se puede entrenar utilizando diferentes combinaciones apagadas, pero como pueden salir infinitas combinaciones lo que se hace es una vez entrenada la red en el proceso de inferencia se utiliza la red original (todas activas) con los correspondientes pesos aprendidos apagando neuronas, y en activación multiplico todos los pesos por [1-p] [probabilidad de haber sobrevivido].

model.add(Dropout(0.5)) [83]

Seguidamente, se define el algoritmo de optimización que se va a utilizar para el descenso por gradiente. En este caso utilizamos *Adam* [84] *(Adaptative Moment Estimation)*, debido a que ha sido el que mejores resultados ha obtenido frente a *Adagrad* [84] *(Adaptative Gradient), y Adadelta* [84] *(an extension of Adagrad)*. Adam es precisamente el más complejo de los 3, y consiste en calcular el learning rate adaptado a cada parámetro. Combina todas las innovaciones introducidas con Adagrad y Adadelta, pero, además, le suma al gradiente el peso anterior, de forma que, si el peso anterior era relevante, tendrá más influencia en el nuevo.

Además, se ha introducido un learning Scheduler. Encontrar un Learning Rate o tasa de aprendizaje adecuado al aplicar descenso de gradiente ayuda a minimizar la función de perdida de una red neuronal. Durante el transcurso del máster se han estudiado una serie de métodos adicionales que pueden hacer que este proceso sea más fluido, rápido y preciso: Learning Rate Schedulers. Estos buscan ajustar el Learning Rate durante la fase de entrenamiento reduciéndolo de acuerdo con un schedule predefinido. Los Learning Rate Schedulers mas sencillos, tal y como se vio en las sesiones teóricas de la asignatura de redes neuronales en el máster, son: **linearly decay (decaimiento lineal)**, y **step decay(decaimiento escalonado):**

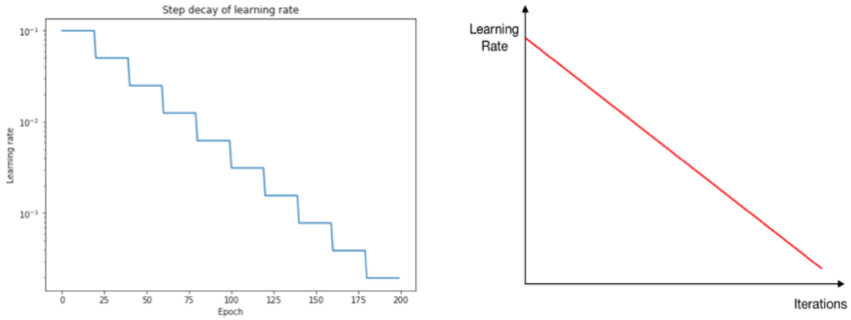

Ilustración 19: Gráficas de un decaimiento escalonado y lineal de la tasa de aprendizaje [85]

Con esta técnica se pretende afinar la búsqueda local. Existen otras técnicas que, en lugar producirse un descenso del learning rate sin retorno, lo que hace es descender hasta un punto donde se produce un reinicio. Va enfriando (annealing) y se producen recalentamientos o reinicios, de forma que se exploran diferentes zonas del espacio de optimización:

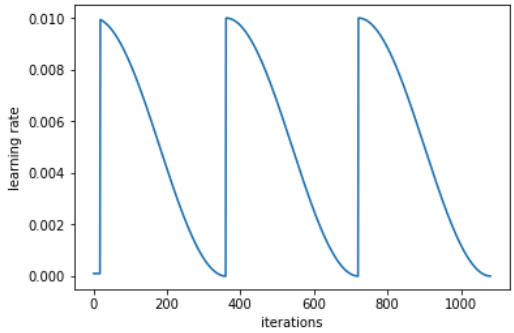

Ilustración 20: Gráfica Tasa de aprendizaje con reinicios [86]

Se trata de **stochastic gradient descent with warm restarts**. Combina un Schedule de enfriamiento agresivo con "restarts" o reinicios periódicos a la tasa de aprendizaje inicial. Se puede definir de la siguiente forma:

$$Lr(t) = Lr_{min} + \frac{1}{2}\left(Lr_{max} - Lr_{min}\right)\left(1 + \frac{Nres}{Ncy} * \pi\right)$$

Donde **Lr(t)** es la tasa de aprendizaje en el instante t, $\mathbf{lr_{min}}$ y $\mathbf{lr_{max}}$ la tasa de aprendizaje minima y máxima respectivamente que fija el usuario, **Nres** el número de epochs desde el último reinicio, y **Ncy** el número de epochs en un ciclo. En el primer periodo de medio coseno se va avanzando hacia un mínimo local en el descenso por gradiente de la siguiente forma:

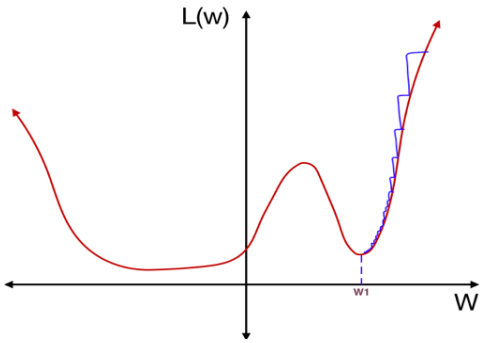

Ilustración 21: Gráfica descenso por gradiente después de un reinicio [85]

Al producirse un recalentamiento "Warm restart" de la tasa de aprendizaje se da un gran paso en la siguiente iteración, y a partir de ese salto se vuelve a descender hacia otro mínimo local durante el segundo periodo del medio coseno. Nuevamente, se aumenta drásticamente dado otro salto. Podría ocurrir que esta vez, cayera en una región más estable de la función de pérdida, este "reinicio" consigue que salga del mínimo local:

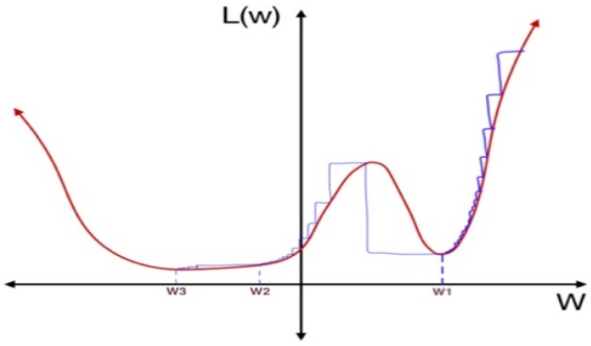

Ilustración 22: Descenso por gradiente con reinicios [85]

Se puede definir el número de ciclos (período del medio coseno), y la duración de cada ciclo para el entrenamiento de la red neuronal. Es recomendable definir un Lr_{max} ligeramente superior al óptimo y suficientemente grande teniendo en cuenta que se esta disminuyendo gradualmente el learning rate y permitiendo que la función salte a un mínimo diferente cuando se "reinicia". Véase el efecto en un modelo 3D:

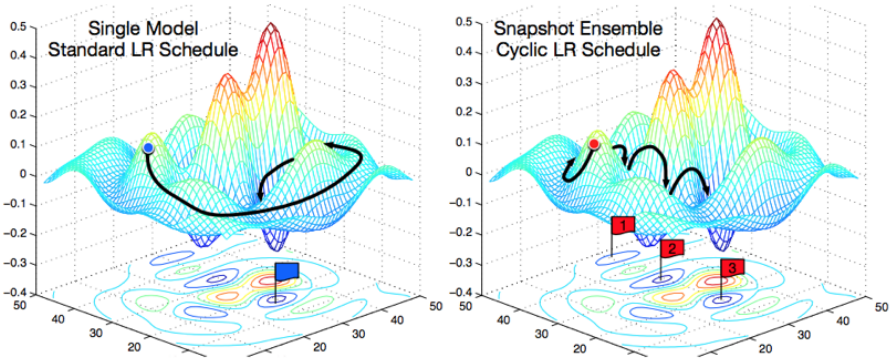

Ilustración 23: Ejemplo funcionamiento de un Lr Schedule sin reinicios Vs. Lr Schedule con reinicios [86]

El modelo de la izquierda desciende lentamente a un mínimo local, mientras que el modelo de la derecha entra y sale de varios mínimos locales, buscando uno más estable. Además de la función del medio coseno, existen otras funciones con las que implementar los reinicios, pero en este trabajo se ha optado por desarrollar esta.

A esta primera propuesta se ha añadido dos nuevas características: En primer lugar, el aumento de la duración del período del medio coseno tras finalizar cada ciclo. En segundo lugar, se ha establecido una reducción del $\boldsymbol{Lr_{max}}$ tras finalizar cada ciclo. Con estas dos modificaciones se pretende que en las primeras epochs se de prioridad a la exploración por las distintas zonas del espacio de búsqueda, y a medida que se avanza en el entrenamiento se vaya aumentando la prioridad de la explotación. De esta forma se pretende conseguir llegar al punto mínimo en una región estable con mayor precisión, y conseguir un mejor resultado:

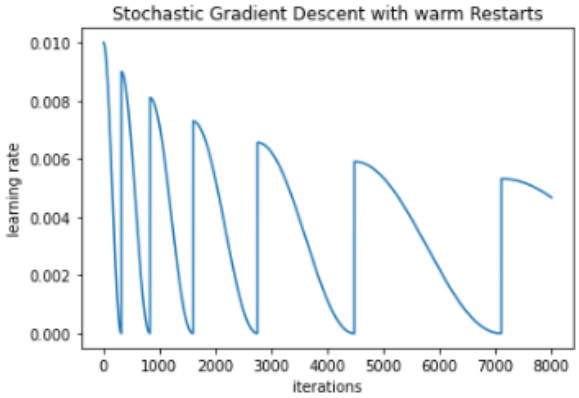

Ilustración 24: Ejemplo gráfica al aumentar el ciclo y disminuir la tasa máxima de aprendizaje (SGDR)

La implementación de la técnica descrita se encuentra en el archivo SGDRScheduler.py Se han ido ajustando los parámetros mencionados con distintas pruebas:

- $Lr_{max} = 1e-2$
- $Lr_{min} = 1e-6$
- *Initial Cycle Size*: 5 epochs
- $Lr_{max}\ descent\ per\ cycle = 0.9$ (reduce en un 10% el Lr_{max} al final de cada ciclo)
- $Cycle\ duration\ increase = 1.5$ (Se incrementa en un factor de 1.5 la duración del periodo del medio coseno en cada ciclo).

Generador de datos

Esta es, seguramente, la parte más compleja de la implementación. Debido a que el tamaño de los datos es bastante grande, debemos establecer un generador que vaya cargando los datos secuencialmente para no sobrepasar el tamaño de memoria. Como ya se ha anticipado, se ha optado por codificar todas las imágenes con la DenseNe121 previamente a hacer el entrenamiento del Decoder. La principal complejidad recae en establecer un entrenamiento mediante "Teacher Forcing" [87].

Se trata de un método para entrenar de forma rápida y eficiente los modelos de redes neuronales recurrentes. Consiste en realimentar la red con el ground truth del instante de tiempo anterior, esto quiere decir que, en entrenamiento, en cada instante de tiempo, en lugar de tener como entrada la salida del instante anterior, se tiene como entrada la verdadera salida del instante anterior. El enfoque clásico para entrenarlos consiste en maximizar la probabilidad de cada token en la secuencia dado el estado actual (recurrente) y el token anterior. En la inferencia, el token anterior desconocido se reemplaza por un token generado por el propio modelo. Esta discrepancia entre el entrenamiento y la inferencia puede producir errores que pueden acumularse rápidamente a lo largo de la secuencia generada

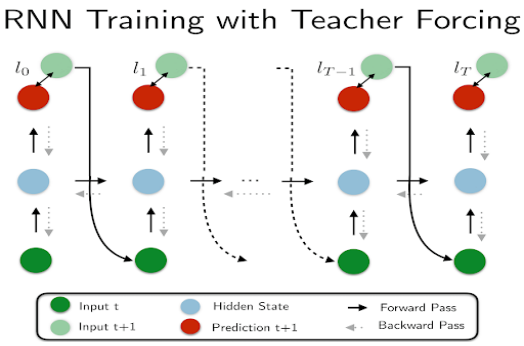

Ilustración 25: Esquema ejemplo de funcionamiento entrenamiento del sistema con Teacher Forcing [88]

En la imagen se puede apreciar claramente como se sustituye la predicción en un instante temporal por el ground truth en ese instante para alimentar la entrada recurrente en el siguiente instante. Todos los detalles de la implementación del generador de datos pueden encontrar en la función **data_generator()** del código en keras.

Además, se ha implementado la función **create_sequence()**, por si se trabaja con un subconjunto más pequeño de datos para poder cargar todos los datos a la vez, en lugar de progresivamente. En este proyecto se ha dispuesto de una máquina Linux con 64 Gb de Ram y dos GPUs y, aun así, ha sido imposible cargar todos los datos a la vez.

Ajuste del modelo

Tras definir las 4 topologías, se procede al entrenamiento de estos para ajustar los pesos al conjunto de datos de entrenamiento. Una vez que los modelos estén ajustados, podremos evaluar la calidad de sus predicciones en el conjunto de datos de pruebas reservado para este fin. Durante el entrenamiento se realiza una monitorización de la calidad del modelo con el conjunto de datos de validación (development). Así, si la calidad del modelo evaluada con el conjunto de datos de validación mejora al final de una epoch, se guarda el modelo como mejor modelo sustituyendo al anterior. Al final de la ejecución, quedará guardado como modelo final, el modelo de mayor calidad obtenido durante el entrenamiento. Podemos utilizar el callback de keras ModelCheckpoint para ello, monitoreando la pérdida mínima en la función de pérdida utilizada "entropía cruzada", con el conjunto de datos de validación. De esta forma también evitamos tener que repetir el entrenamiento desde scratch cuando se interrumpe el entrenamiento por cualquier motivo.

Durante esta fase se han realizado diferentes ajustes. En primer lugar, se redujo el tamaño de secuencia que inicialmente se había fijado en 85 palabras (Tamaño de la descripción más larga), debido a que los entrenamientos se prolongaban durante 4 o 5 días pese a estar trabajando con una máquina potente. Por este mismo motivo se redujo también el tamaño de salida de la LSTM, inicialmente fijado en 256 neuronas a 128 neuronas, con estos dos ajustes, se mejoró el accuracy de los modelos, y se redujo el número de parámetros y el tiempo de entrenamiento a una cuarta parte.

Inicialmente, los pesos de la capa de Embeddings se mantenían congelados durante el entrenamiento. En una segunda prueba, se realizaba un Fine-Tuning de estos pesos utilizando un learning rate fijo más bajo en este segundo entrenamiento para evitar que el ajuste de estos pesos. No obstante, los mejores resultados se han obtenido entrenando desde el inicio con los pesos descongelados.

Con todos estos parámetros ajustados se ha realizado el entrenamiento definitivo de los modelos sin mecanismo de atención:

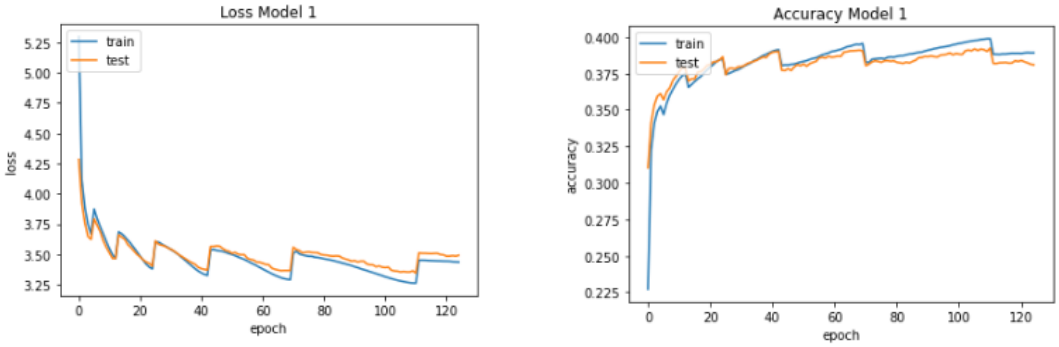

Ilustración 26: Monitorización del entrenamiento de la Topología 1

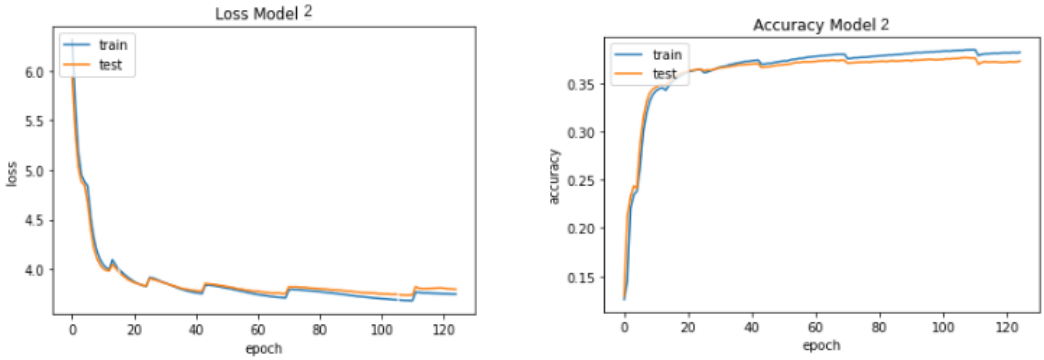

Ilustración 27: Monitorización del entrenamiento de la Topología 2

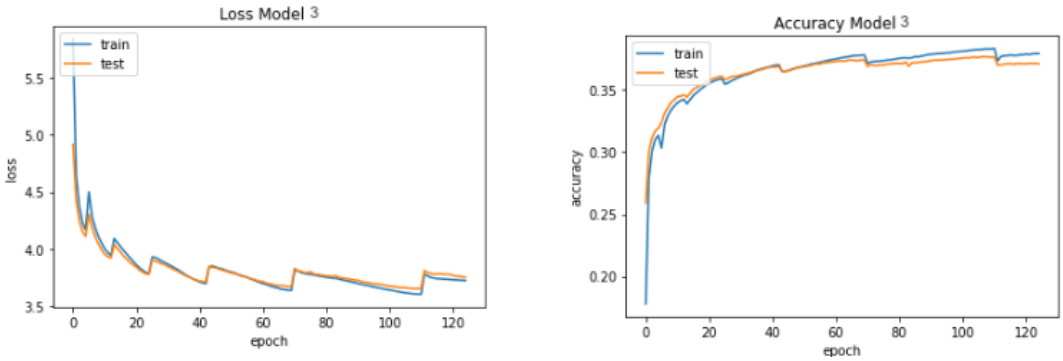

Ilustración 28: Monitorización del entrenamiento de la Topología 3

En los 3 entrenamientos se pueden observar los recalentamientos claramente en las gráficas de accuracy y en las de pérdida. También es evidente que, a medida que pasan las epochs, los ciclos del learning rate son cada vez más largos. Los resultados van mejorando muy parejos, tanto en entrenamiento, como en validación.

En el caso de la cuarta topología que introduce el mecanismo de atención local, ha sido más complejo de monitorear, debido a que se ha definido un entrenamiento ligeramente más complejo de implementar, utilizando funciones de tensorflow algo más complejas. Para ello, se ha seguido parcialmente el siguiente tutorial de tensorflow [89], y añadiendo los detalles de entrenamiento descritos anteriormente.

5. Resultados experimentales y análisis

Tal y como se ha explicado que el accuracy no es una buena medida de calidad para este modelo, y que van a utilizarse otras métricas más apropiadas: BLEU, METEOR y ROUGE-L. La evaluación se hace generando descripciones para todas las imágenes del conjunto de datos de Prueba con cada uno de los modelos ajustados.

Primero, es necesario poder generar una descripción para una imagen usando los modelos. Esto implica inferir de forma similar a lo que se ha hecho la fase de entrenamiento. En este caso, evidentemente, no se aplica la Técnica Teacher Forcing y la LSTM, es alimentada por la salida generada en el instante anterior, en lugar de por el ground truth. Se inicializa la inferencia con el token de inicio de descripción <start>, se continua con el forward por el modelo en cuestión hasta obtener la primera palabra. El proceso continua de manera recursiva, utilizando las palabras generadas como entrada hasta que se alcance el token de final de secuencia <end> o se alcance la longitud máxima de descripción. Se han implementado una serie de funciones imprescindibles para la fase de evaluación e inferencia. Todo este proceso se implementa en la función **generate_desc()**, que dado un modelo entrenado y una imagen genera la descripción del modelo.

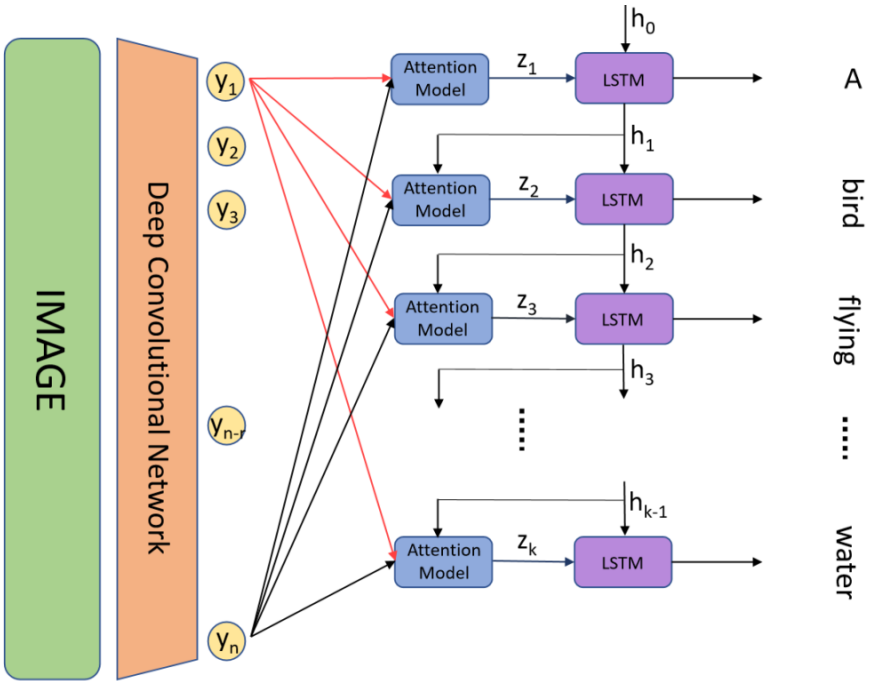

Ilustración 29: Esquema con ejemplo de funcionamiento del sistema generado en fase de inferencia [90]

No se ha mencionado nada de la codificación de imágenes, porque recordemos que conjunto de imágenes de Prueba ya había sido codificado junto al de Entrenamiento o Validación. Sin embargo, para poner el sistema definitivo en funcionamiento con otras imágenes será necesario replicar el proceso de codificación explicado anteriormente para cada nueva imagen. La siguiente función que se presenta es **test_inference().** Esta función recorre todo el Array de imágenes de prueba codificadas generando una descripción (Hipótesis), para cada imagen. Además, devuelve las descripciones proporcionadas en el dataset para esa imagen (Referencias), para poder evaluarlas con las métricas explicadas. Véase los resultados comparados con algunos de los trabajos más mencionados:

Modelo	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L
Topología 1 propia	57	31.8	18.8	11.0	13.1	30.9
Topología 2 propia	54.3	30.3	17.6	10.0	11.9	30.6
Topología 3 propia	55	30.8	18.8	10.7	13.0	30.6
Topología 4 propia (Atención Local)	64.2	45.3	31.4	22.8	18.9	43.1
Donahue et al. **(Codificador-Decodificador)**	58.7	39.1	25.1	16.5	-	-
Jia et al. **(Codificador-Decodificador)**	64.6	44.6	30.5	20.6	17.9	-
You et. al. **(Mecanismo de atención)**	64.7	46	32.4	23	18.9	-
Li and Chen-SR-FL **(Aprendizaje por refuerzo)**	72.0	53.4	38.0	27.1	20.7	48.5
Li and Chen-SR-PL **(Aprendizaje por refuerzo)**	72.9	54.5	40.1	29.3	21.5	49.4

Tabla 6: Comparativa de resultados obtenidos con el conjunto de datos Flickr-30k: topologías propuestas Vs trabajos estado del arte

De la tabla 4 se desprende que los mejores resultados sin aplicar mecanismo de atención se obtienen concatenando la información de la imagen codificada detrás de la LSTM. Realmente, el token de entrada en el instante t=0 siempre será <start>, por lo que, aunque concatenemos la información de la imagen detrás, esta información se replica para cada recurrencia de la red, y le será proporcionada a la LSTM en el instante t=1. Al mismo tiempo, planteándolo de esta forma se evita utilizar la capa repeat_vector y con ello reducimos el numero de parámetros a entrenar.

Sin embargo, se puede observar que, los resultados todavía son muy lejanos a los resultados estado del arte. Sin embargo, añadiendo el modelo de atención local descrito en capítulos anteriores, los resultados mejoran considerablemente, obteniendo mejoras de más de 12 puntos en BLEU de casi 6 puntos en METEOR y de poco más de 12 puntos en ROUGE-L. Pese a haber obtenido resultados todavía inferiores a los resultados estado del arte en esta tarea, hay que tener en cuenta que los trabajos más destacados, tal y como se ha mencionado en la introducción, utilizan técnicas más complejas de Aprendizaje por refuerzo.

Cabe destacar, que el montaje del modelo incluyendo un mecanismo de atención local, y la definición del entrenamiento ha sido mucho más complejo que en las 3 primeras topologías donde se establecía un modelo secuencial o modelo general de keras donde, únicamente se definen las capas del modelo y se realiza el entrenamiento, siendo éste completamente transparente. En este caso, ha habido que definir los diferentes pasos de entrenamiento, funciones de perdida, etc. Siendo todo el proceso mucho más laborioso al implementarse desde tensorflow. Además, se ha realizado una función que plotea las zonas de la imagen donde centra la atención en cada una de las palabras de la descripción generada por el modelo. Véase unos ejemplos de funcionamiento del mejor modelo con imágenes no vistas en entrenamiento:

Ilustración 30: Ejemplo 1: Sistema con Topología propia 4 en fase de inferencia con imágenes de prueba

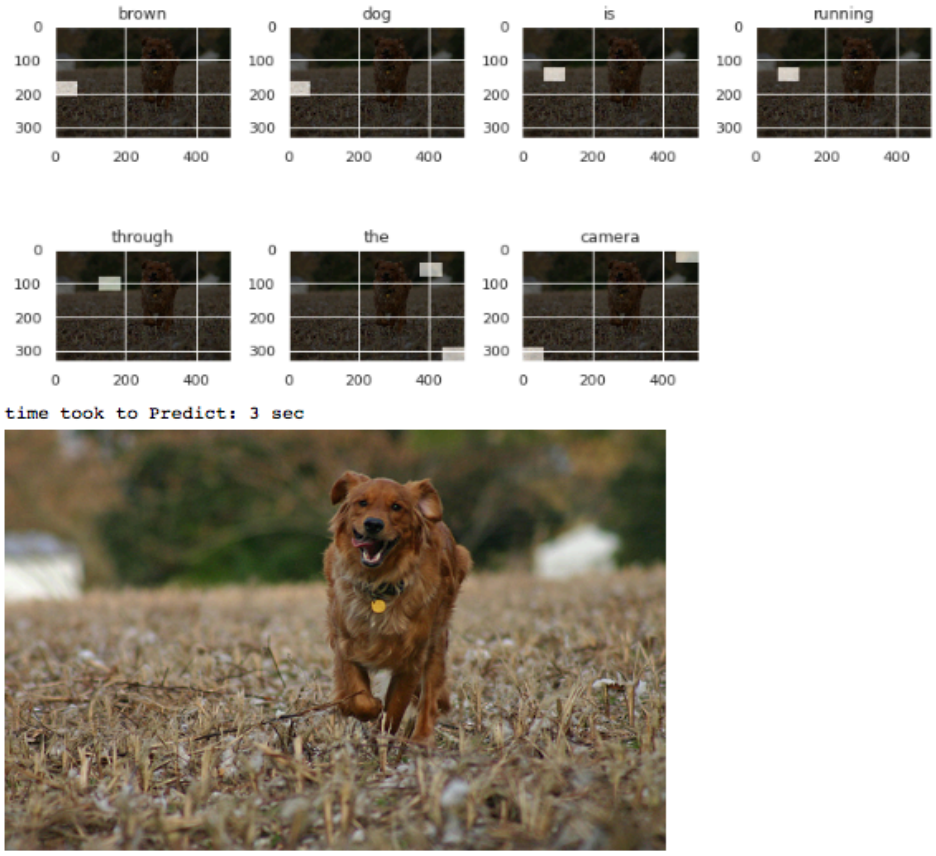

Ilustración 31: Ejemplo 2: Sistema con Topología propia 4 en fase de inferencia con imágenes de prueba

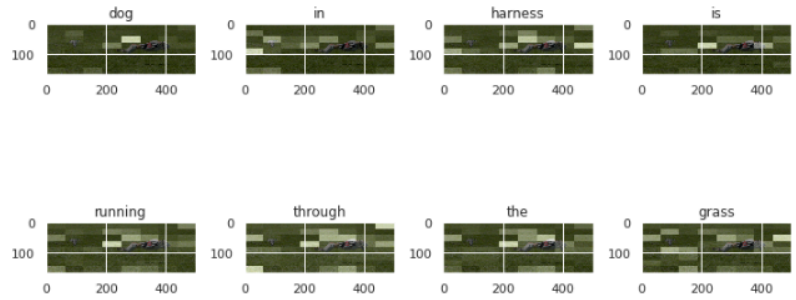

Ilustración 32: Ejemplo 3: Sistema con Topología propia 4 en fase de inferencia con imágenes de prueba

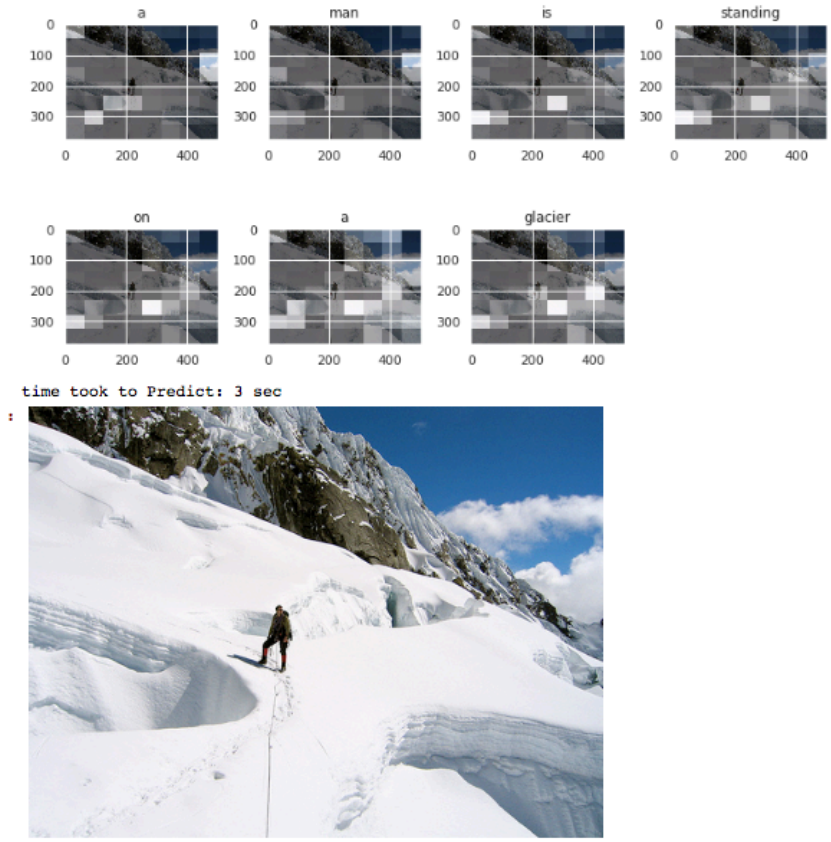

Ilustración 33: Ejemplo 4: Sistema con Topología propia 4 en fase de inferencia con imágenes de prueba

Tal y como se puede observar el sistema generado funciona bastante bien en fase de inferencia, y ya esta listo para la generación automática de subtítulos para imágenes.

6. Conclusiones y trabajos futuros

El objetivo principal de este trabajo era generar un sistema de Image Captioning, para ello ha habido que lidiar con diferentes problemas: elección de enfoque, topologías, y dataset, preparación de los datos, implementación de un generador de datos progresivo para no sobrepasar la memoria de la máquina utilizada, establecer un entrenamiento mediante Teacher Forcing, lo que conlleva establecer un entrenamiento más complejo que (Entrada-target). Es reseñable que se ha trabajado con redes recurrentes que son modelos muy pesados, y ha habido que hacer un ajuste de parámetros para conseguir aliviar al máximo los tiempos de entrenamiento. Además, la implementación del mecanismo de atención local se ha tenido que implementar mediante funciones de tensorflow, de forma que ha habido que desempeñarse con esfuerzo para obtener los resultados esperados Por lo general, ha sido una tarea costosa, pero se ha conseguido sacar adelante con resultados prometedores de cara a un futuro trabajo, donde se implementen nuevas mejoras que permitan acercar los resultados al estado del arte.

Como trabajo futuro se propone la implementación de un modelo que haga uso de Transformers (Modelos de multiatención) en el decodificador. Aunque trabajos recientes [91] proponen incorporar los Transformers tanto en el decodificador, como en el decodificador consiguiendo muy buenos resultados. En este trabajo ya se había realizado una primera implementación con el corpus Flickr-8k para comenzar a ajustar el modelo. Pese a que se reduce la carga en entrenamiento porque son modelos menos pesados que las redes recurrente, no ha sido posible finalizar el ajuste del modelo y transportarlo al corpus de Flickr-30k. Otra opción sería implementar un entrenamiento por refuerzo [92] que es donde se están obteniendo hoy por hoy los mejores resultados para esta tarea.

Bibliografía

[1] M. Hasegawa-Johnson, A. Black, L. Ondel, O. Scharenborg y F. Ciannella, «Image 2 speech : Automatically generating audio descriptions of images,» 2017.

[2] L. Fei-Fei, A. Iyer, C. Koch y P. Perona, «What do we perceive in a glance of a real-world scene?,» *Journal of Vision,* p. 1–29, 31 January (2007), 7(1):10.

[3] J.-Y. Pan, H.-J. Yang, P. Duygulu y C. Faloutsos, «Automatic image captioning,» *Proceedings of the 2004 IEEE International Conference on Multimedia and Expo (ICME 2004),* vol. 3, pp. 1987-1990, 2004.

[4] O. Sargar y S. Kinger, «Image Captioning Methods and Metrics,» de *International Conference on Emerging Smart Computing and Informatics (ESCI)*, Institute of Information Technology, Pune, India, Mar 5-7, 2021.

[5] H. L. Xihui Liu, J. Shao, D. Chen y X. Wang, «Show, Tell and Discriminate: Image Captioning by Self-retrieval with Partially Labeled Data,» *Proceedings of the European Conference on Computer Vision (ECCV),* pp. 338-354, Mar 2018.

[6] B. Dai, S. Fidler, R. Urtasun y D. Lin, «Towards Diverse and Natural Image Descriptions via a Conditional GAN,» *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* p. 2970–2979, Oct 2017.

[7] R. Shetty, M. Rohrbach, L. A. Hendricks, M. Fritz y B. Schiele, «Speaking the same language: Matching machine to human captions by adversarial training,» *Conference: 2017 IEEE International Conference on Computer Vision (ICCV),* vol. 1, pp. 4155-4164, Oct 2017.

[8] R. Vedantam, S. Bengio, K. Murphy, D. Parikh y G. Chechik, «Context-Aware Captions from Context-Agnostic Supervision,» *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* vol. 1, pp. 1070-1079 , 2017.

[9] K. Papineni, S. Roukos, T. Ward y W.-J. Zhu, «Bleu: a Method for Automatic Evaluation of Machine Translation,» *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics,* p. 311–318, Jul 2002.

[10] S. Banerjee y A. Lavie, «METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments,» *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization,* p. 65–72, June 2005; Ann Arbor, Michigan.

[11] C.-Y. Lin, «ROUGE: A Package for Automatic Evaluation of Summaries, pp. 74-81,» de *Text Summarization Branches Out*, Barcelona, Spain, July, 2004.

[12] R. Vedantam, C. L. Zitnick y D. Parikh, «CIDEr: Consensus-based Image Description Evaluation,» de *IEEE Conference on Computer Vision and Pattern Recognition (CVPR); pp. 4566-4575*, June, 2015.

[13] P. Anderson, B. Fernando, M. Johnson y S. Gould, «SPICE: Semantic Propositional Image Caption Evaluation,» *Proceedings of the European Conference on Computer Vision (ECCV),* pp. 382-398, Oct 2016; Amsterdam, the Netherlands.

[14] Wikipedia, «Cross Industry Standard Process for Data Mining,» Wikipedia, 6 Octubre 2020. [En línea]. Available: https://es.wikipedia.org/wiki/Cross_Industry_Standard_Process_for_Data_Mining.

[15] Y. N. Bellini Saibene, M. Volpacchio, S. Banchero y R. Mezher, «Desarrollo y uso de herramientas libres para la explotación de datos de los radares meteorológicos del INTA,» 2014.

[16] J. Brownlee, «Machine Learning Mastery: What is Teacher Forcing for Recurrent Neural Networks?,» Machine Learning Mastery, 6 December 2017. [En línea]. Available: https://machinelearningmastery.com/teacher-forcing-for-recurrent-neural-networks/.

[17] G. Kulkarni, V. Premraj, V. Ordonez, S. Dhar, S. Li, Y. Choi, A. C. Berg y T. L. Berg, «BabyTalk: Understanding and Generating Simple Image Descriptions,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 35, nº 12, pp. 2891-2903, December, 2013.

[18] M. Mitchell, J. Dodge, A. Goyal, K. Yamaguchi, K. Stratos, X. Han, A. Mensch, A. Berg, T. Berg y H. D. III, «Midge: Generating Image Descriptions From Computer Vision Detections,» de *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, pp. 747–756*, Avignon, France, April, 2012.

[19] Y. Yang, C. Teo, H. D. III y Y. Aloimonos, «Corpus-Guided Sentence Generation of Natural Images,» de *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pp. 444–454*, Edinburgh, Scotland, UK., July, 2011.

[20] l. Farhadi, M. Hejrati, M. A. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier y D. Forsyth, «Every Picture Tells a Story: Generating Sentences from Images,» de *Conference: Computer Vision - ECCV 2010, 11th European Conference on Computer Vision, pp. 15-29*, Heraklion, Crete, Greece, September, 2010.

[21] H. Fang, S. Gupta, F. Iandola, R. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. C. Platt, C. L. Zitnick y G. Zweig, «From Captions to Visual Concepts and Back,» *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1473-1482, June 2015; Boston, MA, USA.

[22] C. Zhang, J. Platt y P. Viola, «Multiple Instance Boosting for Object Detection,» *Part of Advances in Neural Information Processing Systems 18 (NIPS 2005),* p. 1417–1424, Jan 2015.

[23] R. Pascanu, C. Gulcehre, K. Cho y Y. Bengio, «How to construct deep recurrent neural networks,» *Computer Science,* 2014.

[24] T. Mikolov, M. Karafiát, L. Burget, J. ". Cernocky y S. Khudanpur, «Recurrent neural network based language model,» *Proceedings of the Eleventh Annual Conference of the International Speech Communication Association, DBLP,* p. 1045–1048, September 2010; Chiba, Japan.

[25] C. Valentini-Botinhao, X. Wang, S. Takaki y J. Yamagishi, «Investigating RNN-based speech enhancement methods for noise-robust Text-to-Speech,» *Proceedings of the 9th ISCA Speech Synthesis Workshop,* p. 146–152, September, 2016; Sunnyvale, CA, USA..

[26] X. Wang, S. Takaki y J. Yamagishi, «An RNN-based quantized F0 model with multi-tier feedback links for text-to-speech synthesis,» *Proceedings of the Interspeech 2017,* p. 1059–1063, August 2017; Stockholm, Sweden.

[27] K. Cho, B. v. Merrienboer, C. Gulcehre, F. B. Dzmitry Bahdanau, H. Schwenk y Y. Bengio, «Learning phrase representations using RNN encoder-decoder for

statistical machine translation,» *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014),* p. 1724–1734, 2014.

[28] Y. Wu, M. Schuster, Z. Chen, J. Dean y etc., «Google's neural machine translation system: bridging the gap between human and machine translation,» *Computer Science,* Sep 2016.

[29] H. Zhang, H. Yu y W. Xu, «Listen, interact and talk: learning to speak via interaction,» *Computer Science,* 2017.

[30] W. Hinoshita, T. Ogata, H. Kozima, H. Kanda, T. Takahashi y H. G. Okuno, «Emergence of evolutionary interaction with voice and motion between two robots using RNN Intelligent robots and systems,» *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems,* p. 4186–4192, October 2009; St. Louis, MO, USA..

[31] Z. Wu, X. Wang, Y.-G. Jiang, H. Ye y X. Xue, «Modeling spatial-temporal clues in a hybrid deep learning framework for video classification,» *Proceedings of the 23rd ACM International Conference on Multimedia,* p. 461–470, October 2015; Brisbane, Australia.

[32] X. Yang, P. Molchanov y J. Kautz, «Multilayer and multimodal fusion of deep neural networks for video classification,» *Proceedings of the 2016 ACM on Multimedia Conference,* p. 978–987, October 2016; Amsterdam, Netherlands..

[33] Z. Wu, Y.-G. Jiang, X. Wang, H. Ye y X. Xue, «Multi-stream multi-class fusion of deep networks for video classification,» *Proceedings of the 2016 ACM on Multimedia Conference,* p. 791–800, October 2016; Amsterdam, Netherlands.

[34] I. Sutskever, O. Vinyals y Q. V. Le, «Sequence to sequence learning with neural networks,» *Advances in Neural Information Processing Systems. 2014,* p. 3104–3112, 2014.

[35] A. Graves, «Generating sequences with recurrent neural networks,» *Computer Science,* 2013.

[36] «Show and tell: a neural image caption generator,» *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* p. 3156–3164, June 2014; Columbus, OH, USA.

[37] S. Hochreiter y J. Schmidhuber, «LONG SHORT-TERM MEMORY,» *Neural Computation 9(8),* pp. 1735-1780, 1997.

[38] K. Cho, B. v. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk y Y. Bengio, «Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation,» *EMNLP,* pp. 1724-1734, June 2014;.

[39] C. Rashtchian, P. Young, M. Hodosh y J. Hockenmaier, «Collecting Image Annotations Using Amazon's Mechanical Turk,» *Association for Computational Linguistics,* vol. Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk, p. 139–147, June 2010, Los Angeles.

[40] H. J. Escalante, C. A. Hernández-Gracidas, J. A. Gonzalez y M. Grubinger, «The segmented and annotated IAPR TC-12 benchmark,» *Computer Vision and Image Understanding,* nº 114, pp. 419-428, April 2010.

[41] M. Hodosh, P. Young y J. Hockenmaier, «Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics,» *Journal of Artificial Intelligence Research 47(1),* p. 853–899, May 2013.

[42] B. A. Plummer, C. M. C. Liwei Wang, J. C. Caicedo, J. Hockenmaier y S. Lazebnik, «Flickr30k entities: collecting region-to-phrase correspondences for richer image-to-sentence models,» *Proceedings of the IEEE International Conference on Computer Vision (ICCV),* p. 74–93, June 2015; Boston, MA, USA.

[43] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollar y C. L. Zitnick, «Microsoft COCO captions: data collection and evaluation server,» *Computer Science,* Apr 2015 .

[44] R. Pascanu, T. Mikolov y Y. Bengio, «On the difficulty of training recurrent neural networks,» *International Conference on Machine Learning,* vol. 52 (3), p. 1310–1318, 2012.

[45] W. Zaremba, I. Sutskever y O. Vinyals, «Recurrent neural network regularization,» *Computer Science ,* 2014.

[46] J. J. L. F.-F. Andrej Karpathy, «Visualizing and understanding recurrent networks,» *Computer Science,* June 2015.

[47] X. Wang, L. Gao, P. Wang, X. Sun y X. Liu, «Two-stream 3D convNet fusion for action recognition in videos with arbitrary size and length,» *Proceedings of the IEEE Transactions on Multimedia,* vol. 20; Issue: 3, pp. 634-644, Mar 2018.

[48] J. Song, H. Zhang, X. Li, L. Gao, M. Wang y R. Hong, «Self-supervised video hashing with hierarchical binary auto-encoder,» *IEEE Transactions on Image Processing,* vol. 27; Issue: 7, p. 3210–3221, 2018.

[49] X. Wang, L. Gao, J. Song y H. Shen, «Beyond frame-level CNN: saliency-aware 3-D CNN with LSTM for video action recognition,» *IEEE Signal Processing Letters,* vol. 24; Issue: 4, p. 510–514, 2016.

[50] I. C. u. VGG16, «Github anunay999,» 2019. [En línea]. Available: https://github.com/anunay999/image_captioning_vgg16.

[51] Y. M. Hironobu, H. Takahashi y R. Oka, «Image-to-Word Transformation Based on Dividing and Vector Quantizing Images With Words,» *"Boltzmann machines", Neural Networks,* pp. 405-409, 1999.

[52] M. Tanti, A. Gatt y K. P. Camilleri, «Where to put the Image in an Image Caption Generator,» *online by Cambridge University Press: 23 April 2018,* vol. 24; Special Issue 3: Language for Images, pp. 467 - 489.

[53] M. Tanti, A. Gatt y K. P. Camilleri, «What is the Role of Recurrent Neural Networks (RNNs) in an Image Caption Generator?,» *Proceedings of the 10th International Conference on Natural Language Generation,* p. 51–60, 2017.

[54] J. Brownlee, «Caption Generation with the Inject and Merge Encoder-Decoder Models,» eep Learning for Natural Language Processing, December 27, 2017. [En línea]. Available: https://machinelearningmastery.com/caption-generation-inject-merge-architectures-encoder-decoder-model/.

[55] V. Mnih, N. Heess, A. Graves y K. Kavukcuoglu, «Recurrent models of visual attention,» *Advances in Neural Information Processing Systems,* nº 3, p. 2204–2212, 2014.

[56] M.-T. Luong, H. Pham y C. D. Manning, «Effective approaches to attention-based neural machine translation,» *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,* p. 1412–1421, Sep 2015.

[57] D. Bahdanau, K. Cho y Y. Bengio, «Neural machine translation by jointly learning to align and translate,» *2014,* Computer Science.

[58] A. M. Rush, S. Chopra y J. Weston, «A neural attention model for abstractive sentence summarization,» *Proceedings of the Conference on Empirical Methods in Natural Language Processing;,* p. 379–389, September 2015; Lisbon, Portugal.

[59] M. Allamanis, H. Peng y C. Sutton, «A convolutional attention network for extreme summarization of source code,» *Proceedings of The 33rd International Conference on Machine Learning, PMLR,* vol. 48, pp. 2091-2100, June 2016; New York, NY, USA..

[60] K. M. Hermann, T. Kočiský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman y P. Blunsom, «Teaching machines to read and comprehend,» *Proceedings of the 28th International Conference on Neural Information Processing Systems,* vol. 1, p. 1693–1701, December 2015; Montreal, Canada..

[61] W. Yin, S. Ebert y H. Schütze, «Attention-based convolutional neural network for machine comprehension,» *Proceedings of the Workshop on Human-Computer Question Answering,* pp. 15-21, June 2016; San Diego, CA, USA.

[62] R. Kadlec, M. Schmid, O. Bajgar y J. Kleindienst, «Text understanding with the attention sum reader network,» *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* pp. 908-918, August 2016; Berlin, Germany.

[63] B. Dhingra, H. Liu, Z. Yang, W. W. Cohen y R. Salakhutdinov, «Gated-attention readers for text comprehension,» *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics,* p. 1832–1846, August 2016; Berlin, Germany.

[64] L. Wang, Z. Cao, G. d. Melo y Z. Liu, «Relation classification via multi-level attention CNNs,» *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics,* vol. 1: Long Papers, p. 1298–1307, August 2016; Berlin, Germany.

[65] P. Zhou, W. Shi, J. Tian, Z. Qi, B. Li, H. Hao y B. Xu, «Attention-based bidirectional long short-term memory networks for relation classification,» *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics,* vol. 2: Short Papers, pp. 207-212, August 2016; Berlin, Germany.

[66] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola y E. Hovy, «Hierarchical attention networks for document classification,» *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* p. 1480–1489, June 2016; San Diego, CA, USA.

[67] J. Song, Y. Guo, L. Gao, X. Li, A. Hanjalic y H. T. Shen, «From deterministic to generative: multi-modal stochastic RNNS for video captioning,» *IEEE Transaction on Neural Networks and Learning System,* vol. 30; Issue: 10, p. 3047–3058, 2018.

[68] J. Song, X. Li, L. Gao y H. T. Shen, «Hierarchical LSTMs with adaptive attention for visual captioning,» *Proceedings IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 42, pp. 1112-1131, may 2020.

[69] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel y Y. Bengio, «Show, attend and tell: neural image caption generation with visual attention,» *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* p. 2048–2057, June 2015; Boston, MA, USA..

[70] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser y I. Polosukhin, «Attention is all you need,» *Proceedings of the 31st International*

Conference on Neural Information Processing Systems, p. 6000–6010, December 2017.

[71] «Effective approaches to attention-based neural machine translation,» *Proceedings of the Conference on Empirical Methods in Natural Language Processing,* p. 1412–1421, September 2015; Lisbon, Portugal.

[72] J. Lu, C. Xiong, D. Parikh y R. Socher, «Knowing when to look: adaptive attention via a visual sentinel for image captioning,» *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* p. 3242–3250, June-July 2016; Las Vegas, NV, USA.

[73] Q. You, H. Jin, Z. Wang, C. Fang y J. Luo, «Image captioning with semantic attention,» *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* p. 4651–4659, June-July 2016; Las Vegas, NV, USA.

[74] T. Gautam, «A Hands-on Tutorial to Learn Attention Mechanism For Image Caption Generation in Python,» Nov 20. [En línea]. Available: https://www.analyticsvidhya.com/blog/2020/11/attention-mechanism-for-caption-generation/.

[75] V. M. Vikram Mullachery, «Image Captioning,» *Computer Science,* May 2018.

[76] P. Ruiz, «Understanding and visualizing DenseNets,» Oct 2018. [En línea]. Available: https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a.

[77] R. Atienza, «2. Deep Neural Networks - 2.4. Densely connected convolutional networks (DenseNet),» de *Advanced Deep Learning with Keras*, Packt, Oct 2018.

[78] J. Pennington, R. Socher y C. D. Manning, «GloVe: Global Vectors for Word Representation,» [En línea]. Available: https://nlp.stanford.edu/projects/glove/.

[79] J. Pennington, R. Socher y C. D. Manning, «GloVe: Global Vectors for Word Representation,» *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP),* p. 1532–1543, 2014; Doha, Qatar; DOI: 10.3115/v1/D14-1162.

[80] J. Mun, M. Cho y B. Han, «Text-guided Attention Model for Image Captioning,» *Computer Science ,* Dec 2016.

[81] Keras, «Layer weight regularizers,» [En línea]. Available: https://keras.io/api/layers/regularizers/.

[82] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever y R. Salakhutdinov, «Dropout: A Simple Way to Prevent Neural Networks from Overfitting,» *Journal of Machine Learning Research ,* vol. 15; Issue: 1, pp. 1929-1958, June 2014.

[83] keras, «Dropout layer,» [En línea]. Available: https://keras.io/api/layers/regularization_layers/dropout/.

[84] S. Ruder, «An overview of gradient descent optimization algorithms,» *Computer Science,* 2016.

[85] C.-F. Wang, «A Newbie's Guide to Stochastic Gradient Descent With Restarts,» Jul 2018. [En línea]. Available: https://towardsdatascience.com/https-medium-com-reina-wang-tw-stochastic-gradient-descent-with-restarts-5f511975163.

[86] M. Hoffmann, «Exploring Stochastic Gradient Descent with Restarts (SGDR),» Nov 2017. [En línea]. Available: https://medium.com/38th-street-studios/exploring-stochastic-gradient-descent-with-restarts-sgdr-fa206c38a74e.

[87] S. Bengio, O. Vinyals, N. Jaitly y N. Shazeer, «Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks,» *Proceedings of the 28th International Conference on Neural Information Processing Systems,* vol. 1, p. 1171–1179, 2015.

[88] C. Lungu, «Training an RNN with teacher forcing,» [En línea]. Available: http://www.clungu.com/Teacher-Forcing/.

[89] Tensorflow, «Image captioning with visual attention,» [En línea]. Available: https://www.tensorflow.org/tutorials/text/image_captioning.

[90] R. Biswas, «Active Learning in Image Captioning,» [En línea]. Available: https://iml.dfki.de/active-learning-in-image-captioning/.

[91] 2. S. C. Wei Liu*1, L. Guo, X. Zhu y J. Liu, «CPTR: FULL TRANSFORMER NETWORK FOR IMAGE CAPTIONING,» *Computer Science,* Jan 2021.

[92] Y. Xiong, B. Du y P. Yan, «Reinforced Transformer for Medical Image Captioning,» *Machine Learning in Medical Imaging, 10th International Workshop, MLMI 2019, Held in Conjunction with MICCAI 2019,* Shenzhen, China, October 13, 2019, Proceedings; .

[93] R. S. C. D. M. Jeffrey Pennington, «GloVe: Global Vectors for Word Representation,» [En línea]. Available: https://nlp.stanford.edu/projects/glove/.

[94] G. H. A. K. I. S. R. S. Nitish Srivastava, «Dropout: A Simple Way to Prevent Neural Networks from Overfitting,» *Journal of Machine Learning Research 15 (2014) 1929-1958,* Mayo 2014.

[95] K. api, «keras dropout,» [En línea].

[96] Z. Yang, X. He, J. Gao, L. Deng y A. Smola, «Stacked Attention Networks for Image Question Answering,» *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* June 2016; Las Vegas, NV, USA.

[97] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft y K. Q. Weinberger, «Snapshot Ensembles: Train 1, Get M for Free,» *Proceedings of 5th International Conference on Learning Representations, ICLR 2017,* Nov 2017.

Declaración de trabajo original

Yo, Pablo Pallarés declaro que soy el autor de este trabajo y que no ha sido copiado o hecho por otras personas.