Universidad Zaragoza
1542

Trabajo Fin de Máster

Sistema de descripción automática de vídeo para la generación de resúmenes audiovisuales

Automatic video description system for the generation of audiovisual summaries

Autor

Eduardo Almazán Galisteo

Director

Eduardo Lleida Solano

Escuela de Ingeniería y Arquitectura

2018

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Eduardo Almázan Galisteo,

con nº de DNI 76923871A en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Máster, (Título del Trabajo) Sistema de descripción automática de vídeo para la generación de resúmenes audiovisuales

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 23 de Agosto de 2018

Fdo: Eduardo Almázan Galisteo

A mi familia

SISTEMA DE DESCRIPCIÓN AUTOMÁTICA DE VIDEO PARA LA GENERACIÓN DE RESÚMENES AUDIOVISUALES

RESUMEN

La gran cantidad de herramientas de búsqueda de documentos multimedia en bases de datos y la dificultad de la indexación de este tipo de contenido, hacen necesaria la existencia de sistemas que permitan agilizar el procesado de gran cantidad de datos para su correcto almacenamiento y uso.

El sistema de descripción automática de video propuesto en este TFM facilita esta tarea. Partiendo de los sistemas ya existentes, basados en indexación descriptiva y añadiendo una capa más que permite combinar la información obtenida de los *frames* gracias a los descriptores utilizados. De esta forma, se atribuye al documento datos de mayor nivel sobre la significación, realizando una combinación entre la indexación semántica y descriptiva. Combinando conceptos de procesado de imagen, redes neuronales y procesado de lenguaje neuronal, se ha generado un sistema dividido en tres bloques diferentes. Estos se encargan de la detección de los *frames* principales a partir de descriptores estadísticos, la descripción de estos a partir de un modelo de *captioning* (descripción de imagen) basado en redes neuronales, y el procesado de las descripciones mediante el análisis de sus *embeddings* y el algoritmo LSA (*Latent Semantic Analysis*).

Una vez desarrollado el sistema, se han explorado diferentes conjuntos de parámetros, con el objetivo de buscar el mejor ajuste posible. Por un lado, se han alcanzado valores de *recall* del 85.9% y 83.13% de precisión para el bloque de selección. Por otro, tras la comparación de diferentes modelos de redes neuronales, se ha obtenido una mejora del 38.76% respecto a los modelos más básicos mediante una red basada en modelos de atención y ajustada mediante SCST (*Self Critical Sequence Training*). Tras esto, se ajustaron los parámetros correspondientes al generador de resúmenes realizando un estudio de los resultados, ya que no existen bases de datos ni métricas con las que compararlos para obtener una cuantificación objetiva del error obtenido.

Finalmente, se analizaron los resúmenes obtenidos para diferentes videos, observando un buen rendimiento general del sistema y destacando la gran variabilidad en el ajuste, provocando que no exista un conjunto de parámetros que permita obtener el mejor rendimiento de forma generalizada, así como la falta de conexión observada entre las frases del resumen debido al único uso del contenido descriptivo de los frames.

Índice general

D. Métricas

Índice de Tablas

Índice de Figuras

Listado de acrónimos

Adam	*Adaptative moment estimation*
API	*Application Programming Interface*
BiLSTM	*Bidireccional Long short-term memory*
BLEU	*Bilingual Evaluation Understudy*
BOVW	*bag-of-Visual-Words*
CBOW	*Continuous Bag of Words*
CIDEr	*Consensus-based Image Description Evaluation*
CNN	*Convolutional Neural Network*
COCO	*Common Objects in Context*
CPU	*Central processing unit*
DFT	*Discrete Fourier Transform*
DTW	*Dynamic Time Warping*
EOS	*End Of Sentence*
FV	*Fisher Vectors*
GloVe	*Global Vectors for Word Representation*
GMM	*Gaussian Mixture Model*
GPU	*Graphics Processing Unit*
HMM	*Hidden Markov Models*
HOG	*Histogram of Oriented Gradients*
HSV	*Hue Saturation Value*
LCS	*Longest Common Subsequence*

LSA	*Latent Semantic Analysis*
LSTM	*Long Short-Term Memory*
METEOR	*Metric for Evaluation of Translation with Explicit Ordering*
MLP	*Multi Layer Perceptron*
MPEG-7	*Moving Picture Experts Group Version 7*
NLI	*Natural Language Inference*
NLP	*Natural Lenguage Processing*
PLSA	*Probabilistic Latent Semantic Analysis*
ReLu	*Rectified Linear unit*
ResNet	*Residual Network*
RGB	*Red Green Blue*
RNN	*Recursive Neural Network*
ROUGE	*Recall-Oriented Understudy for Gisting Evaluation*
RTE	*Recognizing Textual Entailtment*
RTVE	*Radio Televisión Española*
SBD	*Shot Boundary Detection*
SCST	*Self-critcal Sequence Training*
SNLI	*Stanford Natural Language Inference*
SVD	*Singular Value Decomposition*
tf-idf	*Term frequency – Inverse document frequency*
TFM	*Trabajo Fin de Máster*
t-SNE	*t-distributed Stochastic Neighbor Embedding*
UNK	*Unknown*

Capítulo 1

Introducción

En este primer capítulo se ofrece una visión general del proyecto, así como una descripción del concepto y de los objetivos principales. También se detalla la metodología utilizada, así como algunos de sus aspectos básicos.

1.1 Objetivos y alcance

El aumento progresivo de horas de contenido multimedia es un hecho de nuestra realidad, siendo cada vez mayor la cantidad de datos generados. Por ejemplo, en 2016 se registraron una media de 300 horas de video generadas por minuto, así como 3.25 mil millones de horas visualizadas al mes [9]. La gran cantidad de herramientas de búsqueda de documentos multimedia en bases de datos y la dificultad de la indexación de este tipo de contenido, necesitando **hasta 10 veces más que su duración original** para una correcta indexación, hacen muy necesaria la existencia de estos sistemas. Permitiendo agilizar el procesado de gran cantidad de datos para su correcto almacenamiento y su futuro uso.

El **objetivo** de este trabajo fin de máster (TFM) es crear un **sistema capaz de generar de forma automática un resumen** de la longitud deseada describiendo las escenas de mayor importancia de un video. Esto permitirá el etiquetado automático de este y de las escenas que lo componen de forma mucho más rápida. Para esta tarea se han utilizado diferentes **sistemas ya existentes**, así como otras **metodologías que no habían sido aplicadas a esta área**.

Tras realizar un análisis, se han implementado diferentes métodos en código *Python*. Se han utilizado conceptos de **procesado estadístico** de imagen, descripción automática mediante la utilización de **redes neuronales** y procesado de las descripciones mediante métodos de **procesado de lenguaje natural**. De esta forma, es posible realizar de forma autónoma una tarea como la indexación explícita de video, ahorrando gran cantidad de recursos e indexando una mayor cantidad de datos en menor tiempo.

1.2 Concepto

Este proyecto se compone de **tres bloques diferentes**, que se han ido modelando progresivamente y enlazando conforme iban siendo desarrollados. El sistema se puede dividir en las siguientes partes:

- **Muestreador de *frames* principales:** Toma el video de entrada, realiza un muestreo de **n** *frames* (fotogramas) por segundo y elige cuales de estos *frames* poseen la relevancia suficiente para ser descritos. El sistema se basa en un detector de cortes de escena, que

hace uso de una comparación por histogramas de color y una comparación por histogramas de gradiente orientado. De esta forma es posible detectar cuando ocurre un cambio significativo entre escenas y así obtener el *frame* central, o los *frames* intermedios además del central, para ser descritos en el siguiente bloque.

- **Descriptor automático de imagen**: Basado en una red neuronal ***Residual Network*** **(ResNet)** y una **red neuronal recursiva (RNN)**, se encarga de tomar los *frames* principales y obtener su descripción, permitiendo procesar la información visual y convirtiendola en información semántica que será procesada en el último de los bloques.
- **Generador de resúmenes:** El generador de resúmenes toma las descripciones generadas anteriormente y realiza un procesado de estas mediante una red neuronal ***Bidireccional Long Short-Term Memory*** **(BiLSTM)** para obtener la representación vectorial de estas (*embedding).* Con estos *embeddings* es posible hallar la relación entre cada una de las descripciones, discriminando aquellas con una similitud demasiado alta, ya que no aportan nueva información, y aquellas con una relación demasiado baja. Finalmente, se procederá a la generación del resumen mediante el algoritmo ***Latent Semantic Analysis*** **(LSA)**.

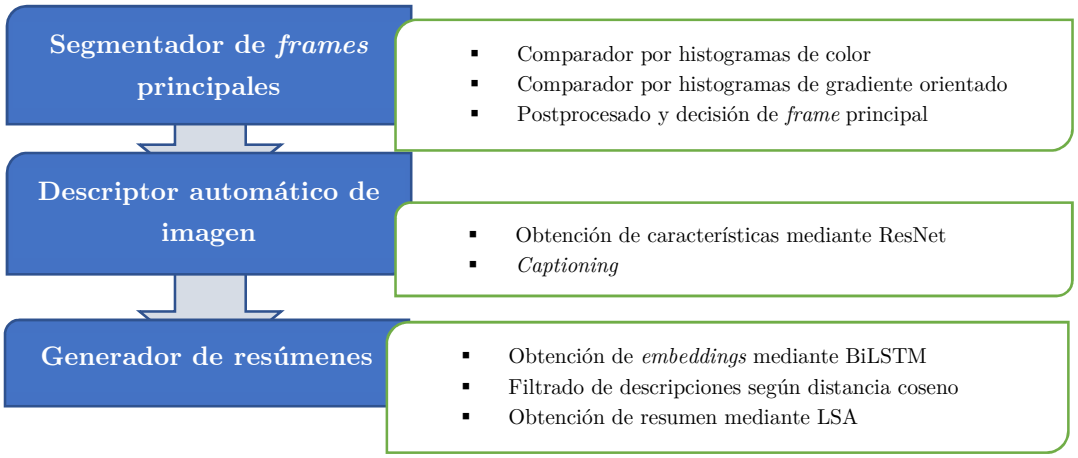

Figura 1.1: Composición del sistema

Finalmente, con el sistema implementado y tras realizar diversas correcciones y mejoras, se procederá a la obtención de diferentes tipos de resumen, variando los parámetros del sistema para ver cómo afectan al resultado final.

1.3 Metodología

Entre las herramientas utilizadas para el desarrollo de este proyecto se encuentran los diferentes modelos utilizados, implementados en lenguaje de programación ***Python*** [7]. *Python* es un lenguaje de programación interpretado, multiplataforma y orientado a objetos. Estos factores, junto a su licencia de código abierto, permiten una **gran flexibilidad** para el desarrollo de programas, funciones y librerías, pudiendo encontrar una gran cantidad de funciones previamente creadas en continuo desarrollo y actualización.

En este caso, para el uso y entrenamiento de las diferentes redes neuronales, se han utilizado dos sistemas. Uno inicial, generado mediante la biblioteca *Keras* [6] desarrollada también en *Python* y montada sobre *TensorFlow*, en el que se realizaron las primeras pruebas. Finalmente, una red generada sobre la biblioteca *PyTorch* [8], de mayor complejidad computacional pero también con mejores resultados y un aumento en la velocidad de entrenamiento y cálculo, basado en la red *NeuralTalk* [3]. Ambos modelos utilizados se basan en códigos ya desarrollados [1] [2] para el entrenamiento y evaluación en la **base de datos *Common Objects in Context* (COCO)** [5] [10], de los que hablaremos más adelante. También se ha elegido *PyTorch* para la implementación de la red neuronal que permitirá clasificar y relacionar las descripciones [4], y de este modo poder procesarlas para alcanzar el resultado final.

Con todos estos recursos, podemos analizar los diferentes pasos por los que ha pasado el proyecto, y como estos han sido ejecutados y solucionados:

1. **Recolección** de bibliografía e información sobre los actuales sistemas de descripción de video, así como de los fundamentos teóricos a implementar en el sistema.
2. **Estudio de los principales conceptos** y de los bloques que compondrán el sistema final, así como el estudio del funcionamiento de los conceptos básicos de Python, como de las librerías a utilizar.
3. **Implementación** del bloque de segmentación de *frames* principales mediante la comparación de Histogramas de Color e Histogramas de Gradiente Orientado.
4. **Estudio** del código utilizado para la generación del **modelo de descripción** de imágenes. Puesta en marcha, corrección y mejora.
5. **Entrenamiento** de diferentes modelos de redes neuronales y comparación de métricas para la obtención del mejor modelo posible. Adaptación del modelo para trabajar con los *frames* obtenidos del segmentador anteriormente desarrollado.
6. **Estudio** del código utilizado para la **generación de *embeddings*** a partir de las descripciones obtenidas y la utilización de la relación entre estos para la generación del resumen. Representación e interpretación de la información obtenida.
7. **Implementación del generador** de resúmenes. Adaptación al sistema de detección y descripción.
8. Implementación de **mejoras y corrección** de errores en los tres bloques desarrollados. Comprobación de resultados y del sistema final.
9. Generación de resúmenes para diferentes tipos de video. Interpretación y análisis de los **resultados** obtenidos. Análisis de las causas de estos.
10. Creación de la **memoria final**. Esta fase y la anterior se realizarán de forma paralela, ya que una vez finalizada la fase de desarrollo y obtenidos los resultados experimentales es necesario plasmarlos en la memoria.

1.4 Organización de la memoria

Además de este capítulo en el que se encuentra una pequeña introducción de los aspectos iniciales del proyecto a modo de resumen, esta memoria contiene 4 capítulos más y 4 anexos complementarios:

- **Capítulo 2 - Estado del Arte:** Un pequeño resumen de los modelos y sistemas actuales para la descripción automática de video además de las técnicas principales en las que estos se basan.
- **Capítulo 3 - Sistema de descripción automática de video:** Se describirá como han sido implementados los distintos bloques, así como la interconexión entre estos.
- **Capítulo 4 - Trabajo experimental y resultados:** Recopilación de los resultados de cada una de las partes del sistema, así como la consideración de los resultados finales obtenidos.
- **Capítulo 5 - Conclusiones y Líneas futuras:** Se describirán las conclusiones a las que se han llegado tras la finalización del proyecto, así como las diferentes líneas de mejora que podrían implementarse en un futuro.
- **Anexo A – Descriptores Estadísticos:** Contiene una explicación detallada de los descriptores utilizados.
- **Anexo B – Redes Neuronales:** Contiene una explicación más detallada sobre el concepto de redes neuronales y sus diferentes tipos.
- **Anexo C – Procesado del Lenguaje Natural:** Explica con mayor detalle las técnicas de procesado de lenguaje natural utilizadas en este proyecto.
- **Anexo D – Métricas:** Detalla los fundamentos teóricos en los que se basan las métricas para la medición de la calidad del modelo de *captioning*.

Capítulo 2

Estado del Arte

En este capítulo se describe el estado actual de las tecnologías y metodologías relacionadas con la descripción automática de video. Se describen los intereses principales, así como de forma general las técnicas más utilizadas.

2.1 Introducción

El objetivo actual del reconocimiento de video es el análisis automático de la actividad y la descripción automática de secuencias de video para su indexación y posterior búsqueda y adquisición.

Este problema está recibiendo una gran atención en el área de procesamiento de imágenes y visión artificial. De hecho, el uso generalizado de los medios digitales, la aparición de formatos de vídeo cada vez más compactos y la disminución de los costos de almacenamiento, ha generado un aumento exponencial en cuanto a la cantidad de datos multimedia generados y almacenados. Sin embargo, para que estos datos puedan ser utilizables de forma eficaz, es necesaria su clasificación, para que puedan ser vistos a través de un catálogo [11]. Esto permite dividir el problema de la clasificación e indexación en dos vertientes diferentes: la **indexación explícita**, en la que es necesaria que un operador especializado se encargue de forma manual del etiquetado, y la **indexación automática**, en la que un algoritmo es capaz de extraer mediante diversos métodos la información de la pista de video y proceder a su etiquetado de forma autónoma

2.2 Indexación Explícita

La indexación explícita consiste en la **clasificación manual** de los documentos y de los archivos audiovisuales. Esta clasificación debe ser hecha por un operador especializado que atribuya al documento datos de alto nivel sobre la significación del contenido. Las consultas normalmente generadas son palabras asociadas a un objeto, una acción, el nombre de un personaje o un acontecimiento en concreto. Esta asignación hace que normalmente, la indexación explicita se considere por naturaleza **indexación semántica.**

Sin embargo, el hecho de realizar esta operación de forma manual es una tarea demasiado larga, pudiendo necesitar **hasta 10 veces la duración** de la secuencia, y acarreando otros problemas como que los datos interesantes en una fecha determinada no lo fueran necesariamente en la fecha de indexación. Por ello, el análisis automático de documentos facilita en gran medida el trabajo del operador. Utilizando métodos que ayudan a simplificar y agilizar el trabajo, generando una combinación entre la indexación explícita o manual y la automática:

- **Preselección** de grandes bases de datos de imágenes.
- Indexación **semi-automática** con interacción operador/computadora.
- División de vídeo en planos y **simplificación** en "imágenes clave."

2.3 Indexación Automática

La indexación automática consiste en el análisis autónomo del contenido audiovisual para su clasificación o etiquetado. Mientras que la indexación explícita suele ser de naturaleza semántica, la indexación automática es **esencialmente descriptiva o visual**. El algoritmo de indexación atribuye datos de bajo nivel semántico, sobre el contenido geométrico o espectral de la imagen a un nivel local o global. Esto hace que las consultas se realicen, por ejemplo, por modelo, realizado una comparación con las características almacenadas en la base de datos obtenidas tras el procesado, escogiendo aquellas que compartan una mayor verosimilitud.

Es posible encontrar una gran cantidad de metodologías, entre las que se pueden destacar dos: las **aproximaciones de una sola capa**, es decir, utilizando únicamente la secuencia de imágenes que componen el video para la detección y el reconocimiento, y en segundo lugar la **aproximación jerárquica**, donde se representan las acciones de alto nivel con la composición de acciones más sencillas. Siendo necesario el uso de técnicas de **predicción estructurada** para la detección de la secuencialidad de los eventos. Algunos ejemplos de estas técnicas pueden ser modelos probabilísticos como los **Modelos Ocultos de Markov (HMM)** o las **gramáticas intercontextuales** [13].

En cuanto a las aproximaciones de una capa, se observan dos subtipos: las **secuenciales**, que plantean el vídeo como una secuencia de observaciones, también basadas en **HMM** y otras técnicas como el **Alineamiento Temporal Dinámico (DTW)**, y las **espaciotemporales**, que se encargar de extraer las características de la imagen mediante el uso diferentes técnicas basadas en descriptores. Más concretamente, en la última aproximación de este grupo, que es en la que se basaran los sistemas del proyecto, se encuentran las **características locales espaciotemporales**. Las principales ventajas de estas técnicas son su **robustez ante el ruido y las variaciones presentes** en los entornos realistas, por ejemplo: cambios de luz, entornos dinámicos, etcétera. Sin embargo, no presentan un buen funcionamiento para reconocer tareas complejas compuestas por diferentes eventos, ya que se pierde la secuencialidad. Para la extracción de las características más relevantes de la imagen, el método más utilizado es **el uso de diferentes tipos de descriptores** [13], entre los cuales es posible diferenciar:

- **Descriptores de color:** Espacio de color, cuantización de color, análisis de capas, análisis de estructura de color, colores dominantes…
- **Descriptores de textura:** Histograma de bordes, homogeneidad de texturas, diferenciación…
- **Descriptores de forma:** Análisis de contornos, análisis de regiones, análisis tridimensionales…
- **Descriptores de movimiento:** Análisis de movimiento, movimiento paramétrico, trayectorias, movimiento de cámara…
- **Descriptores de localización:** Localizador de regiones, localizador espacio temporal…

Muchos de estos descriptores se encuentran implementados en el estándar de video **MPEG-7**, sin embargo, no existen gran cantidad de procesos definidos por **MPEG-7** donde el usuario tenga libertad de actuación debido a la necesidad de expansión del estándar [14].

Por su gran capacidad de computación y resultados, las redes neuronales [15] también se encuentran en uso en el campo de procesado de imagen. Siendo las más comunes las **redes neuronales convolucionales** (**CNN**) en combinación con capas lineales. Permiten una mayor extracción de las características de la imagen, siendo capaces de obtener unos mejores resultados al detectar y clasificar los elementos presentes como veremos más adelante. Sin embargo, para la utilización de estas técnicas es necesaria una gran cantidad de recursos computacionales: *clusters* de **CPU**, **GPU** de última generación y una programación orientada al paralelismo sobre estos recursos hardware.

Una vez obtenidas las características locales necesarias, es posible realizar una **codificación** de estas mediante diferentes técnicas para la aplicación de conceptos como el **reconocimiento de formas** y el **aprendizaje automático**. Entre las técnicas más utilizadas, se puede destacar, debido a su sencillez y efectividad, la representación ***bag-of-Visual-Words (*BOVW*)***. Utiliza conceptos como el *clustering* para representar mediante un histograma de frecuencias las características más cercanas a cada término representado en la imagen. También es posible encontrar técnicas de mayor complejidad como pueden ser la combinación de los modelos generativos y discriminativos con los denominados **Fisher Vectors (FV)**, en los que se caracterizan las muestras en base a su desviación con un **modelo generativo de mixtura Gaussiana (GMM)**.

Finalmente, tras un correcto procesado, extracción y clasificación de las características de la imagen, es posible la identificación de las diferentes entidades que componen la imagen y de la relación entre estas, que serán utilizadas para la indexación del documento:

- **Elementos de la imagen:** elementos identificables en la imagen a los que se les pueda asignar una etiqueta identificable, como pueden ser objetos, personas, animales…
- **Fondos:** Define el ambiente en el cual se encuentran los demás elementos identificados, puede ser un lugar, un clima…
- **Gestos:** Movimientos llevados a cabo por el cuerpo humano o los elementos detectados en la imagen, como “levantar una mano”, “sonreír”.
- **Acciones:** Movimientos llevados a cabo por una sola persona o elemento compuestos de gestos como “caminar” o “nadar”.
- **Interacciones:** Contempla actividades que involucran a dos o más personas o elementos de la imagen, como “amigos hablando”.
- **Actividades de grupos:** Se compone de uno o varios conjuntos de personas y/u objetos llevando a cabo una actividad como “un grupo de caballos corriendo”.

Una vez identificadas las entidades que componen el video, es posible realizar consultas, por ejemplo, realizando **búsquedas por elemento** o utilizando **imágenes semejantes**.

En el sistema desarrollado se añadirá una capa más, combinando la información obtenida de los *frames* gracias a los diferentes descriptores y el sistema de redes neuronales. Se atribuirá al documento **datos de mayor nivel sobre la significación** del contenido que los obtenidos únicamente mediante indexación descriptiva, realizando así una

combinación entre ambos tipos de indexación. Esto permitirá búsquedas asociadas a palabras o descripciones en sistemas de indexación automática.

2.4 Sistemas comerciales

Existen diversos tipos de sistemas comerciales dedicados al análisis de video y a la obtención de datos de este. Muchos de estos sistemas **ajustan sus funciones** para un fin en concreto, como puede ser la detección de **personas** y **movimiento** en sistemas de video seguridad o la detección de **eventos** para sistemas de análisis deportivo. En el caso del análisis más general del video y de sus elementos, los dos sistemas de mejor rendimiento actualmente en el mercado son ***Cloud Video Intelligence*** [57] de ***Google Cloud*** y ***Video Indexer*** [58], desarrollado por ***Microsoft Azure.***

Cloud Video Intelligence permite realizar búsquedas en los vídeos y ayuda a su indexación, ya que extrae sus **metadatos** con una API (*Application programming interface*). *Cloud Video Intelligence* realiza anotaciones rápidamente en los vídeos almacenados y ayuda a **identificar entidades clave** (nombres) en el vídeo, así como el momento en el que aparecen. Identifica qué es importante y qué no recuperando la información relevante de todo el vídeo, por toma o por fotograma.

Video Indexer es una aplicación en nube creada con *Azure Media Analytics*, *Cognitive Services* y *Azure Search*. Le permite **extraer información** de sus vídeos mediante tecnologías de **inteligencia artificial**. Entre sus funciones desarrolladas, *Video Indexer* es capaz de realizar detección automática del idioma, transcripción, detección facial, indexación de hablantes, detección de fotogramas, traducción, moderación y etiquetado.

Ambos sistemas realizan funciones similares a las que se desarrollaran en este proyecto, sin embargo, tan **solo proporcionan como resultado los elementos** obtenidos mediante el análisis del video. En este proyecto se realizará un **procesado de esos elementos** para la obtención de descripciones más concretas y la generación de un resumen explicativo a partir de estas.

2.5 Fundamentos Teóricos

Para el desarrollo del sistema serán necesarios el estudio y análisis de las diferentes ideas. De esta forma es posible realizar una **subdivisión de los principales conceptos** utilizados en cada una de las fases del proyecto:

- **Descriptores estadísticos:** principalmente los **histogramas de color** y los **histogramas de gradiente orientado** para la detección de los *frames* principales en la primera etapa del proyecto. También se hace uso de otras técnicas como **comparación por la transformada discreta de Fourier (DFT)** o un algoritmo simple para la generación de los umbrales de decisión.
- **Redes Neuronales:** El sistema está compuesto tanto por **CNN** como por **RNN**. En los siguientes apartados se detallarán sus conceptos básicos, pudiendo encontrar información más detallada en el **Anexo A**. Estas redes se encuentran presentes tanto en el **descriptor automático de imagen** como en la **generación de *embeddings***.

- ***Embeddings*:** Generados mediante una red **BiLSTM,** permitirán **cuantificar la relación entre las descripciones** obtenidas y discriminar aquellas que no añadan información relevante.
- **Análisis semántico latente:** Genera el resumen final a partir del algoritmo **LSA,** una técnica de procesado de lenguaje natural, en particular **semántica distribuida**, que permite la obtención de aquellas descripciones con mayor peso. Estas, de número variable, son las que compondrán el resultado final.

2.5.1 Detección de cambio de escena

Como base del análisis de contenido y del procesado de video, la **detección de cambio de escena (SBD)** juega un importante rol en el manejo de la información. Una escena es cada parte del documento audiovisual que transcurre en un mismo espacio y en un mismo tiempo. Estos cambios de escena son divididos en cambios abruptos y graduales. Los **cambios abruptos** son aquellos en los que dos escenas diferentes conectan de forma directa sin necesidad de edición, mientras que los **graduales** transcurren mediante la sucesión de diferentes *frames* modificados por programas de edición. Debido al gran porcentaje de cambios abruptos sobre los cambios graduales, este proyecto se especializará en estos, debido a su mayor simplicidad y a la obtención de mejores resultados [16].

Para la detección de estos cambios, se utilizarán **dos tipos diferentes de descriptores**: los **histogramas de color**, y los **histogramas de gradiente orientado** (**HOG**). Los histogramas de color permitirán la comparación de los **espacios de color** entre los *frames* adyacentes. Los HOG servirán como método de respaldo al anterior, basándose en la **detección de objetos** mediante la identificación de su gradiente, ya que la magnitud de estos se dispara en los cambios abruptos de intensidad. De esta forma, será posible identificar los diferentes cambios mediante la medida de **discontinuidad entre *frames*** adyacentes. Este **procedimiento se encuentra detallado en el Anexo A**.

2.5.2 Redes Neuronales

Las redes neuronales [18] son una herramienta de **modelado no paramétrica**. Están basadas en el aprendizaje de funciones complejas mediante la adaptación de sus parámetros en base a ejemplos, con el objetivo de tomar límites de decisión **no lineales**, que mejoren la probabilidad de acierto respecto a los lineales.

A pesar de no ser un concepto relativamente moderno, puesto que los primeros modelos datan de la década de los 40, las redes neuronales, a raíz de su mayor automatización, han supuesto un **avance muy significativo** para la tecnología en un gran número de campos. Por ello, entorno al 2010, se revivió el interés por este tipo de técnicas, siendo en la actualidad una de las más aplicadas debido a su flexibilidad y buenos resultados. Entre los tipos de redes neuronales, se hará uso de **dos diferentes**, las redes neuronales **convolucionales** (CNN) y las **recurrentes** (RNN).

El mayor cambio en la arquitectura de las CNN consiste en la utilización de **imágenes como entrada**:

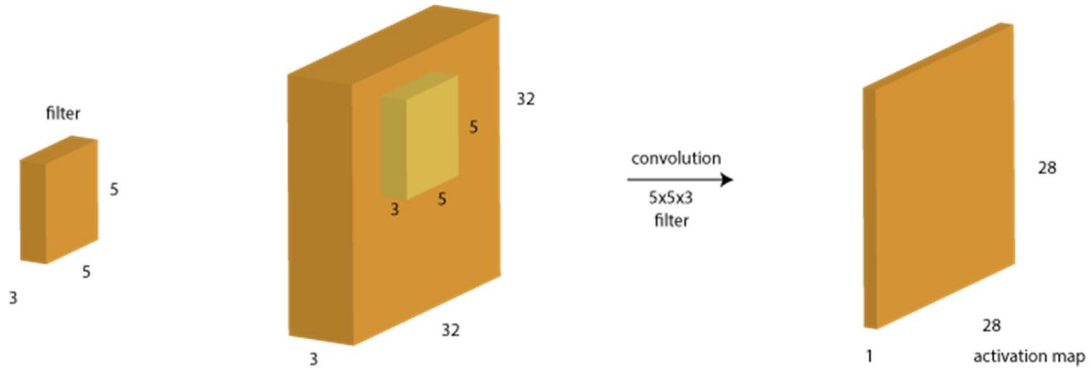

Figura 2.1: Ejemplo de redes neuronales convolucionales [55]

Esto permite realizar ciertas modificaciones a su arquitectura, incrementando la eficiencia de la implementación y reduciendo enormemente la cantidad de parámetros que manejar. Toman ventaja del hecho de que la **entrada consiste en imágenes y modifican su arquitectura** en base a esto. En particular, las capas de una red neuronal convolucional tienen neuronas distribuidas en **3 dimensiones: ancho, alto y profundidad (anchura, altura y capas de color)**. Para evitar la saturación y degradación de la precisión provocada por la profundidad de estas redes, se utilizarán redes ResNet, que solucionan el problema mediante el uso del **aprendizaje residual**. Esto permitirá obtener redes funcionales de **mayor profundidad y complejidad**.

Las redes **neuronales recurrentes (RNN)** [22] son un tipo de redes con **bucles**, lo que permite que la **información persista**, es decir, como el razonamiento ante eventos previos puede servir para los posteriores. Esto las hace unas herramientas muy potentes en el **aprendizaje de secuencias**. En concreto, se utilizarán las redes **LSTM *(Long Short Term Memory),*** así como su versión **bidireccional**. Este es un tipo especial que funciona, en muchas ocasiones, mejor que la versión estándar. Son capaces de asimilar **dependencias de larga duración** y una gran ayuda a la hora de manejar **grandes cantidades de datos.** Los fundamentos sobre los que se basan estas redes, así como algunos ejemplos de ellas se pueden encontrar de forma detallada en el **Anexo B.**

2.5.3 *Word Embeddings*

En la actualidad, gran cantidad de algoritmos de procesado y muchas de las arquitecturas de *machine learning* and *deep learning* son **incapaces de procesar cadenas de texto** o texto plano en su forma original. Estos sistemas requieren de **conjuntos de números** como entrada para poder ser capaces de ejecutar cualquier tipo de trabajo, como pueden ser problemas de clasificación, clusterización, regresión… Para procesar la gran cantidad de datos que se encuentra en formato texto, es necesario un método que permita extraer y utilizar este conocimiento.

En términos simplistas, los ***word embeddings*** [25] [26] [27] son la **representación numérica de una palabra**, conjunto de palabras o de un texto. Generalmente se encargan de **mapear** una palabra perteneciente a un diccionario, a un vector. Algunas de sus propiedades y sus métodos de generación se encuentran explicados de forma más detallada en el **Anexo C**.

2.5.4 Análisis Semántico Latente (LSA)

El **análisis semántico latente (LSA)** [28] es una técnica de procesado lenguaje natural (**NLP**), perteneciente a la **semántica distribuida**, que permite analizar la relación entre un conjunto de documentos y los términos que los componen. *LSA* asume que las palabras cercanas en su significado ocurrirán en fragmentos similares del texto.

Entre sus diferentes usos, se encuentra la **generación de resúmenes**, en el cual se centrará este proyecto. Existe la posibilidad de utilizar como entrada las descripciones generadas y obtener la relación entre estas, pudiendo seleccionar las que posean una **mayor relevancia** entre ellas para la generación del resultado.

Sin embargo, LSA posee algunas **limitaciones**: la interpretación de los **valores negativos** obtenidos tras la descomposición SVD (*Singular Value Decomposition*), la imposibilidad de capturar la **polisemia** de las palabras o el **desajuste del modelo probabilístico** utilizado. LSA asume que las palabras y los documentos analizados forman un modelo gaussiano conjunto (hipótesis ergódica), mientras que se ha observado que, de forma general, la distribución se ajusta más a una distribución de *Poisson*. Para **mayor detalle** de este método, consultar el **Anexo C** de la memoria.

Capítulo 3

Sistema de descripción automática de video

En este capítulo se analizarán, paso a paso, los bloques desarrollados para la generación del sistema de descripción automática de video, basados en los fundamentos teóricos explicados en el anterior capítulo y en los anexos.

3.1 Introducción

El sistema de descripción automática de video para la generación de resúmenes se encuentra compuesto por **un conjunto de tres bloques diferenciados** conectados uno tras otro. Cada uno de estos bloques tendrá una función diferenciadora en el conjunto del sistema, y serán explicados con detalle en los siguientes apartados:

- **Detector de *frames* principales:** Se encarga de la **obtención de los *frames* principales a partir del muestro** de los *frames* del video y su consiguiente análisis basado en **descriptores estadísticos (Histograma de color e Histograma de gradientes orientados)**. Tras el muestreo y el diezmado, se analizará la estadística de la señal generada a partir de la diferencia entre los descriptores de cada muestra para proceder a su selección. Finalmente se revisarán mediante el **espectro**, obtenido mediante DFT, de los fotogramas seleccionados.
- **Descripción automática de *frames* principales:** Toma los *frames* obtenidos del bloque anterior y genera la descripción de estos. Para ello, utiliza una red **CNN ResNet de 101 capas** y una **red especializada en *captioning*** (generación de descripción de imagen) compuesta principalmente por una **red RNN LSTM**. Estas redes han sido entrenadas mediante la base de datos COCO y generadas mediante la utilización de **modelos de atención** y **SCST (*Self-critcal Sequence Training*)**.
- **Generación del resumen mediante NLP:** Procesa las descripciones y obtiene su **representación en un espacio vectorial** mediante *embeddings* generados a partir del algoritmo *Infersent,* el cual se encuentra basado en una red neuronal BiLSTM. Permite obtener las relaciones entre estas mediante la **distancia coseno**. Después se realizará un **filtrado** para eliminar aquellas que se encuentren repetidas o no tengan suficiente similitud con el resto. Por último, se obtendrá el **resumen final** a partir de las más representativas mediante el uso del algoritmo LSA.

3.2 Detector de *frames* principales

El primer bloque del sistema se encargará de realizar la detección de los cambios de escena en el video, y la posterior **obtención de los planos** de mayor importancia.

Para ello, se hará uso de los **descriptores** analizados en el **anexo A**, así como de diversas técnicas de **postprocesado** para mejorar la detección de cortes y el cálculo de los **umbrales** en función de la estadística de los *frames* analizados. Se procederá a una **comparación** de los *frames* obtenidos **mediante DFT** para la eliminación de aquellos que correspondan a la misma escena y hayan sido elegidos por error. Por último, se añadirá un **sobremuestreo** en la detección del *frame* principal. Mediante este sobremuestreo variable, se consigue reducir la posibilidad de que el *frame* elegido como principal esté movido o no contenga la información más descriptiva de la escena.

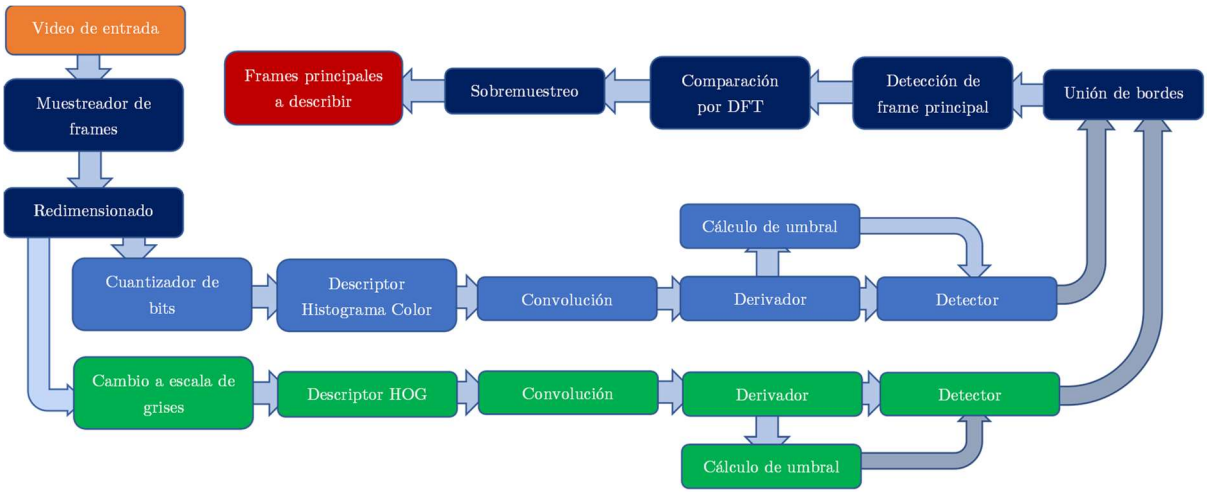

Figura 3.1: Composición del detector de *frames* principales

3.2.1 Muestreo y redimensionado

Tras tomar el video de entrada, se realizará un muestreo de este, tomando de forma uniforme **n** *frames* por segundo. Cuanto mayor sea el número de *frames* muestreado mayor será el tiempo de procesado, pero también se obtendrá una menor diferencia entre los *frames* adyacentes durante las escenas, **mejorando los resultados** obtenidos posteriormente.

La resolución y el tamaño de las imágenes muestreadas es **habitualmente demasiado grande** en términos de procesado de imagen. Por ello, tras el muestreo se llevará a cabo un redimensionado de estas que **disminuya la cantidad** de información a analizar y elimine a su vez información que resulte **redundante** para los algoritmos. Esto nos permite una **gran disminución en el tiempo de procesado**, tomando habitualmente valores de redimensionado de 2, 4 e incluso 8 en ambos ejes, siendo estos elegidos en función del tamaño de la entrada. Tras este redimensionado se procederá de forma paralela a la obtención de ambos descriptores para la detección de los cortes entre las escenas.

3.2.2 Detección mediante Histograma de Color

Antes de realizar el cálculo de las diferencias mediante el descriptor, se realizará una disminución del número de bits del espacio de color. Usualmente, en imágenes digitales **RGB** se poseen **24 bits por píxel**, repartiendo estos 24 bits en 8 para cada componente. El histograma definido estará compuesto por un vector de **M** elementos, donde **M** es el número de colores posibles en el espacio de color. Si realizamos la computación de forma

directa, se obtiene que el número de colores posible es exageradamente alto (2^{24}) incrementando enormemente el tiempo de computación. Debido a la respuesta limitada del ojo, el ser humano es incapaz de distinguir todos los niveles de color. Por lo tanto, una solución simple consistirá en considerar tan solo los **bits más significativos** de cada uno de los componentes RGB.

Figura 3.2: Cuantización de los 8 bits RGB [30]

Mediante esta cuantización, es posible reducir el tamaño de los histogramas a 2^{12} colores (4096), **reduciendo así la cantidad de memoria y tiempo necesarios**. La eliminación de bits será el primer proceso realizado a los *frames* nada más ser muestreados, tras lo cual pasaran al generador de histogramas y a su comparación.

Tras la generación de la diferencia mediante la medida de la discontinuidad entre planos adyacentes, se **procederá a un postprocesado** de esta, debido a que los resultados obtenidos experimentalmente no se aproximan tanto a la función delta ideal. El primer paso consiste en una **convolución** de la señal diferencia obtenida con una **ventada rectangular** de tamaño **M.**

$$\boldsymbol{HistDif_{conv}[i] = HistDif[i] * \frac{1}{W} \cdot rect\left(\frac{i}{W}\right)} \qquad \textbf{(3.1)}$$

Con esta operación de procesado de señal se **suaviza** la señal **HistDif[i]**, de forma que las pequeñas variaciones debido al cálculo de las diferencias desaparecen. Sin embargo, mantenemos las características principales de los cortes para poder detectarlos más tarde. Los cortes abruptos pasaran de un modelo ideal tipo delta a un **modelo tipo ventana rectangular**:

$$\boldsymbol{HistDif_{conv}^{cut}[i] = \frac{\alpha_i}{W} \cdot rect\left(\frac{i - i_{cut}}{W/2}\right)} \qquad \textbf{(3.2)}$$

Tras la implementación y comprobación del bloque, se han estudiado diferentes valores para la anchura de la ventana, escogiendo finalmente $\mathbf{W = 20.}$ Una vez suavizada la señal de diferencias, se procederá a la detección de los cortes, intentando identificar el patrón rectangular en la señal convolucionada.

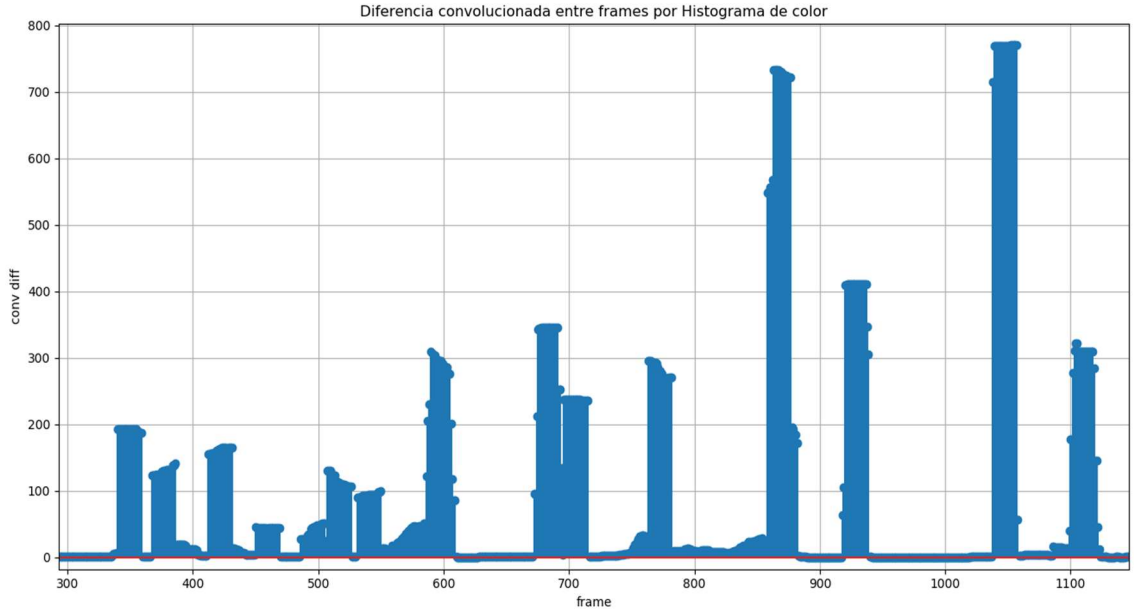

Figura 3.3: Convolución entre HistDif y pulso

Un método muy efectivo que **reduce la distorsión** producida por el movimiento de objetos y cámara es la aplicación de **la primera derivada** a la señal $\mathbf{HistDif_{conv}[i]}$**:**

$$\boldsymbol{HistDif_{conv}^{deriv} = [1, -1] * HistDif_{conv}} \tag{3.3}$$

Como se puede observar en la siguiente gráfica, al aplicar la primera derivada sobre la señal convolucionada el patrón de ventanas se transforma en **un pico positivo seguido de un pico negativo**, ambos espaciados una distancia de **W** muestras, siendo **W** el tamaño de la ventana rectangular utilizada en la convolución.

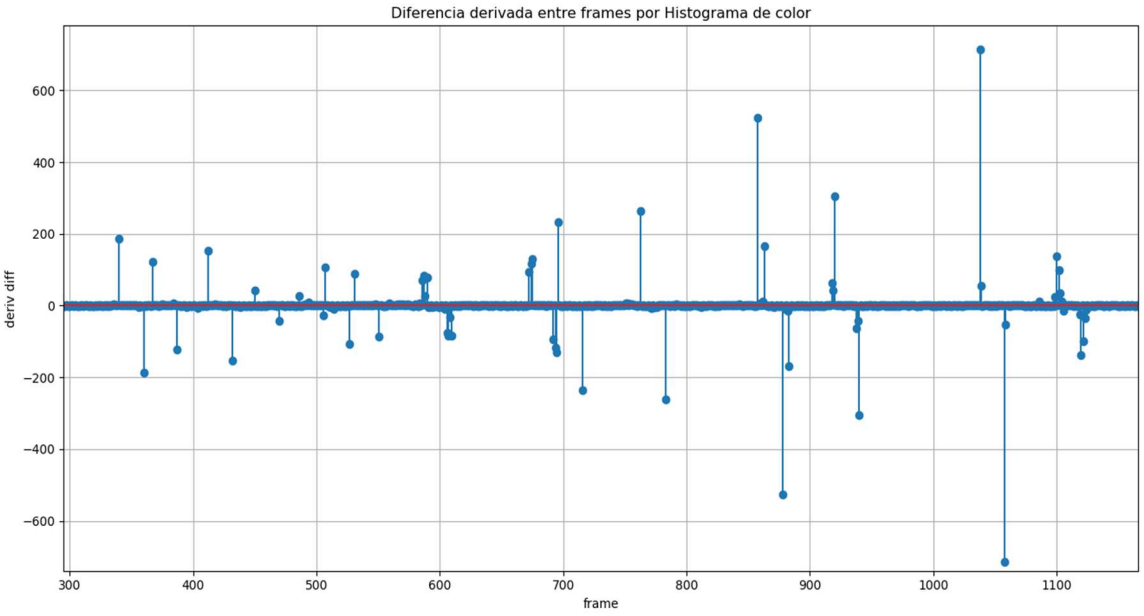

Figura 3.4: Primera derivada de la señal convolucionada

Un *frame* que cumpla esta condición, a la vez que **ambos picos excedan cierto umbral** calculado posteriormente, se considerará como un corte en la secuencia de video. Específicamente, el ***frame* posicionado entre el pico positivo y el negativo** se tomará como comienzo de la nueva escena, que coincide con el centro de la señal rectangular encontrada tras la convolución. De esta manera, los cortes serán detectados si una variación abrupta conlleva a un cambio porcentual significativo en la variación de color de la secuencia de video.

3.2.3 Detección mediante Histograma de Gradiente Orientado (HOG)

De forma paralela a la detección por histograma de color, se realizará la detección por histograma de gradiente orientado, combinando posteriormente los resultados de ambos. El **proceso es generalmente el mismo** que en el caso anterior. Se tomarán los *frames* muestreados, realizando una conversión a escala de grises, y calculando los descriptores basados en **HOG** como esta explicado en el **anexo A** y se hallará la discontinuidad entre *frames* adyacentes.

Esta medida de discontinuidad también será **convolucionada por una ventana rectangular** del mismo tamaño que en el caso anterior, obteniendo el patrón de picos positivo y negativo mediante la **aplicación de la primera derivada** [30] para la obtención del *frame* de corte.

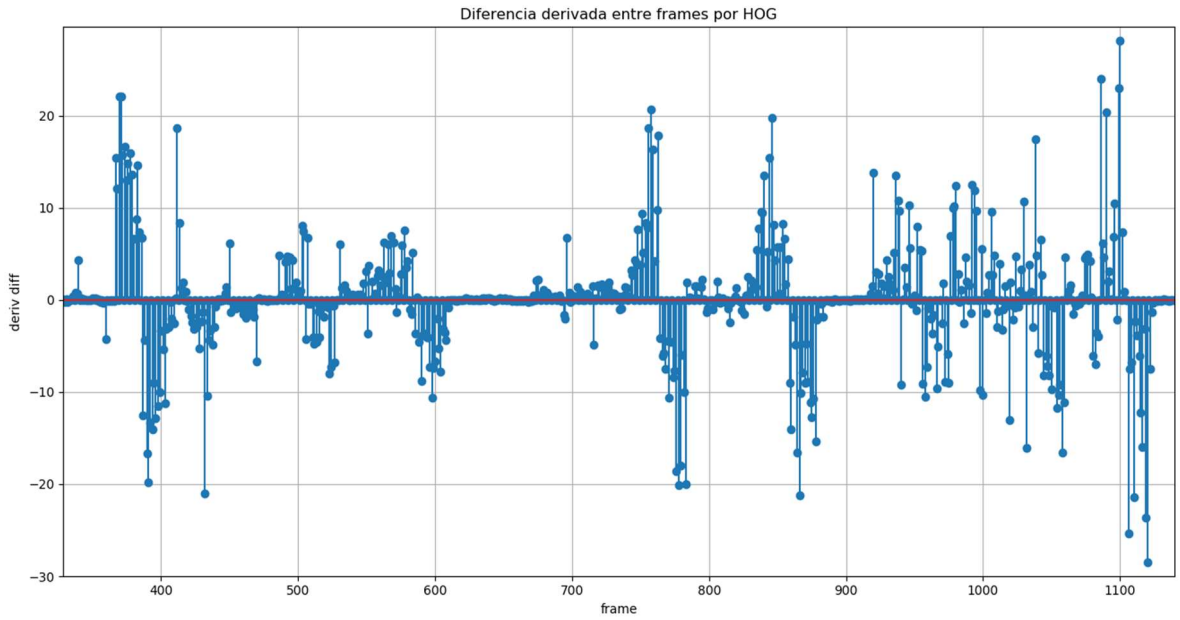

Figura 3.5: Primera derivada de la señal convolucionada HOG

Se puede observar como los cortes detectados por HOG poseen una **mayor cantidad de ruido y distorsión** que en el caso anterior, siendo esta técnica utilizada de forma **auxiliar**. Estos factores se tendrán en cuenta a la hora de generar los umbrales de decisión que marcarán las muestras en las que se detectarán las transiciones.

Tras la realización de diversas pruebas, es evidente que la utilización de ambos descriptores concentra la **mayor carga del sistema,** como se verá más adelante, debido a la gran cantidad de *frames* a procesar y el tamaño de estos. Por ello es necesario un **compromiso** entre el tiempo de procesado y la calidad de los resultados. Un **menor valor de muestreo** en el número de *frames* o un **diezmado** demasiado elevado de la imagen, aunque puedan **reducir considerablemente** los tiempos de ejecución, también influyen en gran medida en los resultados obtenidos, pudiendo ser estos errores propagados por los demás bloques del sistema.

3.2.4 Selector de *frame* principal

Una vez obtenidas las salidas de los derivadores en ambas ramas, se procederá a la decisión de los *frames* principales. A partir de este punto, **ambas señales se tratarán paralelamente**, realizando las mismas operaciones en ambas, con alguna modificación en los parámetros de ajuste de las funciones como única diferencia.

En primer lugar, se tomarán las salidas del derivador para proceder al cálculo de su media y varianza. De esta forma se obtendrá un umbral que dependa de la señal de entrada, el cual podrá ser ajustado según el parámetro variable **S**, modificándolo según el tipo de video. Para la rama de detección por color se ha seleccionado $\mathbf{S = 2}$ mientras que para la rama HOG se utiliza $\mathbf{S = 4}$.

$$\boldsymbol{m} = \frac{\sum_{j=1}^{N-1} \boldsymbol{Dif}_{\boldsymbol{conv}}^{\boldsymbol{deriv}}[\boldsymbol{j}]}{\boldsymbol{N-1}} \tag{3.3}$$

$$\boldsymbol{\sigma} = \sqrt{\sum_{\boldsymbol{j=1}}^{\boldsymbol{N-1}} \frac{\left(\boldsymbol{Dif}_{\boldsymbol{conv}}^{\boldsymbol{deriv}}[\boldsymbol{j}] - \boldsymbol{m}\right)^2}{\boldsymbol{N-1}}} \tag{3.4}$$

$$\boldsymbol{T} = \boldsymbol{m} + \boldsymbol{S} \times \boldsymbol{\sigma} \tag{3.5}$$

Una vez calculado el umbral se procederá a la **detección** de los *frames*. En primer lugar, se localizarán aquellas muestras que **rebasen el umbral** tanto de forma positiva o negativa, para detectar ambos picos resultantes de la derivación de la ventana de detección. Una vez todos los picos se hayan localizado, se calculará el **punto medio** de cada par de picos, obteniendo la posición de los *frames* para cada uno de los dos métodos.

Tras la detección inicial de todos los *frames*, se procederá a la **unión de ambas ramas**. Para ello, se establecerá mediante parámetro una **distancia mínima entre *frames*** para habilitar una detección de cambio de escena. De esta forma se evitarán errores que pudieran ser producidos por desajustes entre ambos métodos y se limitará la duración mínima de las escenas. Cambios excesivamente de rápidos pueden no ser interesantes al no aportar información relevante, de alta complejidad en la descripción o *frames* demasiado distorsionados. Debido a que el método por histograma de **color** se considera el **método principal**, se dará **prioridad** a los *frames* detectados por este, seleccionando estos en caso de no coincidencia entre ambos métodos. La distancia mínima seleccionada varía según el número de *frames* **n** muestreados por segundo y de la longitud del video analizado, siendo $\mathbf{min_{DIFF} = 1.5 \cdot n}$ aproximadamente.

Por último, tras obtener el vector de cortes, al cual se le añadirán tanto el *frame* inicial como el final, se calculará el *frame* principal de cada escena **como el intermedio entre cada par de cortes**. Debido a la imposibilidad de conocer que *frame* contiene la mayor cantidad de información o es el más representativo sin un **análisis mucho más exhaustivo y costoso**, se ha tomado la decisión de tomar el intermedio debido a su simplicidad. Sin embargo, se intentará limitar los errores que pueda generar esta decisión mediante el último bloque del sistema, el **sobremuestreo de *frames* principales**.

3.2.5 Comparación mediante DFT

Ya con los *frames* principales localizados, se realizará una comparación de la similitud de estos, que permita eliminar aquellos **excesivamente parecidos** que hayan sido extraídos por error. De esta forma se evitarán repeticiones de escenas no deseadas. Para ello se procederá del mismo modo que en los casos anteriores.

En primer lugar, se realiza una conversión a escala de grises del *frame* para disminuir la complejidad del sistema y el tiempo de computación. Tras esto se calcula la **DFT bidimensional** de todos los *frames* marcados como principales por el paso anterior y se procede al cálculo de la **diferencia entre *frames* adyacentes de igual forma** que en el cálculo de las diferencias por Histograma de color y por HOG, utilizando para ello la **ecuación 3.6**.

$$\boldsymbol{DFTDif[i] = \sum_{n=1}^{N}\sum_{m=1}^{M} \frac{(DFT_i - DFT_{i-1})^2}{DFT_i + DFT_{i-1}}} \qquad \textbf{(3.6)}$$

siendo $\mathbf{DFT_i}$ el resultado de la DFT bidimensional del *frame* **i** en escala de grises, y **N, M** las dimensiones de este. De esta forma se obtendrá el vector de diferencias DFT, del que extraer el umbral mediante las **ecuaciones 3.3, 3.4 y 3.5**. Por último, se definirá un factor $\mathbf{F_{DFT}}$ por el que se escalará el umbral para disminuirlo. Todos los *frames* cuya diferencia se encuentre por debajo de este umbral serán eliminados. Para el umbral de detección correspondiente con el bloque DFT se utiliza $\mathbf{S = 0}$ y $\mathbf{F_{DFT} = 0.6}$.

Comparando el uso de los histogramas de color con la DFT para comprobar que método se adecuaba más, se ha elegido realizar la comparación mediante el **espectro** de las imágenes. Esto es debido a la **similitud de los resultados** obtenidos entre ambos métodos, y a las ventajas que puede aportar el uso de la DFT en casos en los que la distribución cromática de dos planos continuos sea muy parecida. De esta forma se podrán cubrir **errores que no sea capaz de solucionar el descriptor por histograma** de color, contando con las **ventajas proporcionadas por ambos métodos**.

Tras **obtener los *frames* principales definitivos**, se extraerá la posición temporal de los comienzos y finales de cada escena y de sus *frames* principales. Estas marcas temporales serán combinadas con las descripciones obtenidas por el descriptor automático para la **generación de subtítulos**. Así se podrá realizar una **comprobación mucho más eficiente** del resultado mediante la combinación de ambos bloques, además de la posibilidad de aprovechar estos subtítulos en un futuro para otros fines.

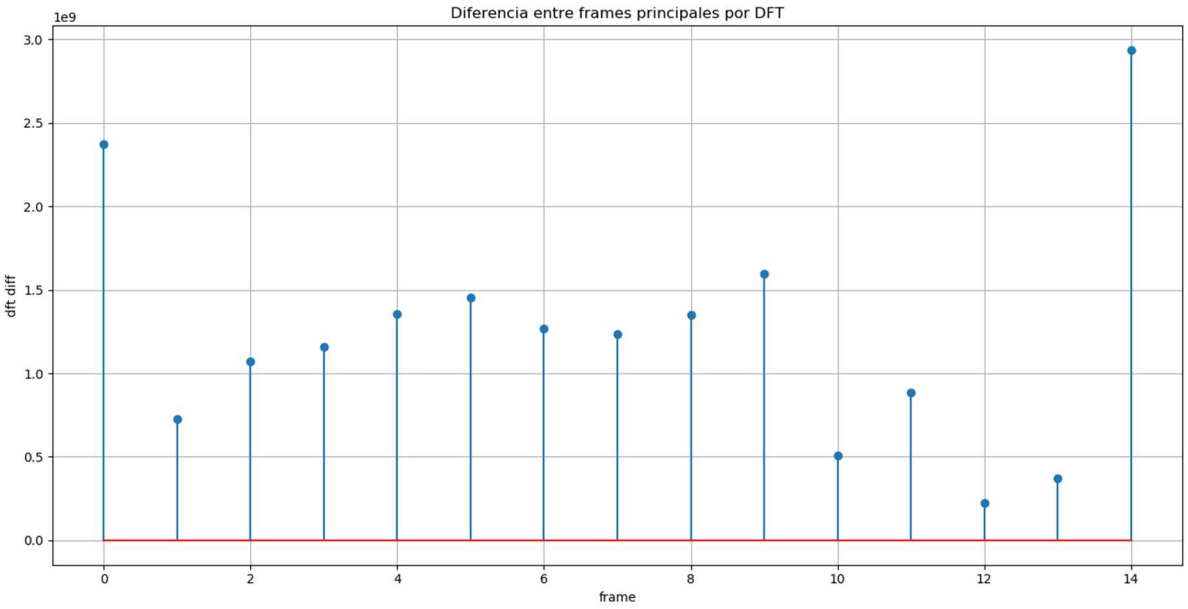

Figura 3.6: Diferencias entre las DFTs de los *frames* seleccionados

3.2.6 Sobremuestreo de *frames* principales

El último bloque del sistema realizará un **sobremuestreo opcional** de los *frames* principales. Debido a la imposibilidad de conocer que punto de la escena que contiene el *frame* de más información, una vez detectado su principio y final, se ha tomado la decisión de extraer el *frame* intermedio. Para mejorar el rendimiento del sistema e intentar corregir los errores en los que **este *frame* no fuera el más correcto** para la descripción, se tomará **más de un *frame* de cada escena**.

De esta forma, se aumenta la probabilidad de escoger el *frame* correcto a costa de un pequeño incremento en el tiempo de procesado. Además, la extracción de ***frames* repetidos se verá corregida más adelante** al realizar el **filtrado de las descripciones** en la última fase del sistema. Allí, los *frames* repetidos y las descripciones excesivamente parecidas serán eliminadas, de igual manera que las descripciones de *frames* movidos que no tengan un grado de relación suficiente con el resto. Normalmente, se toma un valor $\mathbf{FR_{UPSAMPLE} = 3}$, tomando tres *frames* por escena.

3.3 Descripción automática de *frames* principales

El segundo bloque del sistema se encargará de tomar los *frames* seleccionados por el bloque anterior y **obtener una descripción** de cada uno de ellos, también llamada *caption*. Mediante el uso de una red CNN ResNet se **extraerán las características** de las imágenes contenidas en la base de datos COCO. Mediante estas y la información adicional proporcionada por la base de datos, se ajustarán los ficheros de configuración que permitirán el **entrenamiento de la red** de descripción. Esta se basa en una **red LSTM en combinación con otras capas** más simples. De esta forma, se entrenará un modelo con el cual poder **evaluar las imágenes** obtenidas mediante el extractor de *frames* principales. Finalmente, se extraerán todas las descripciones obtenidas en formato texto para ser enviadas al último bloque del sistema.

Figura 3.7: Composición del descriptor automático

3.3.1 Base de datos COCO

La **base de datos COCO (*Common Objects in Context*)** [31] introduce un nuevo conjunto de datos que permite abordar los **tres problemas principales** presentes en el entendimiento de imágenes: detección de objetos en perspectivas no canónicas, razonamiento contextual entre objetos y la localización precisa de estos.

- **Perspectivas no canónicas**: Para todas las categorías de objetos, existe una versión canónica. Los sistemas de reconocimiento habituales son capaces de detectar estas perspectivas, pero tienen **problemas en situaciones de la vida real**, en la que estos objetos pueden estar semi ocultos o en el fondo, lastrando el rendimiento de estos sistemas.
- **Razonamiento contextual entre objetos**: La identificación múltiple de objetos tan solo puede ser resuelta en muchas ocasiones mediante un **razonamiento contextual entre estos**, debido a su pequeño tamaño o su apariencia ambigua. Para este razonamiento, es necesario el tratamiento de la **escena en conjunto** en vez de los objetos en solitario.
- **Localización**: El entendimiento y la **detección detallada de la posición** de los objetos es un componente fundamental del análisis de la escena. Esta ubicación puede definirse mediante delimitadores o máscaras a nivel de píxel.

COCO contiene **91 categorías** de objetos diferentes, de las que 82 de ellas poseen más de 5000 instancias etiquetadas. En total, la base de datos posee aproximadamente **2.5 millones de instancias etiquetadas** repartidas en **328.000 imágenes**. En contraste con otras bases de datos como ImageNet, COCO contiene un menor número de categorías, pero más instancias por categoría. Además, uno de los puntos diferenciadores es el número de instancias etiquetadas en comparación con el resto, las cuales ayudan al aprendizaje de la información contextual. La base de datos COCO tiene **tres niveles de extracción** diferentes:

- **Clasificación de imágenes**: Requiere de la presencia de etiquetado binario que **indiquen la presencia de objetos** en la imagen.
- **Detección de objetos**: Combina la clasificación de imágenes con la **localización de la posición exacta** de objetos mediante el uso de delimitadores.
- **Etiquetado semántico de escenas**: Requiere que cada píxel de la escena se etiquete como perteneciente a una categoría. A diferencia de la detección, no necesita la localización de las instancias, lo que permite **etiquetados de instancias que serían difícilmente definibles** (hierba, calle, muro…)

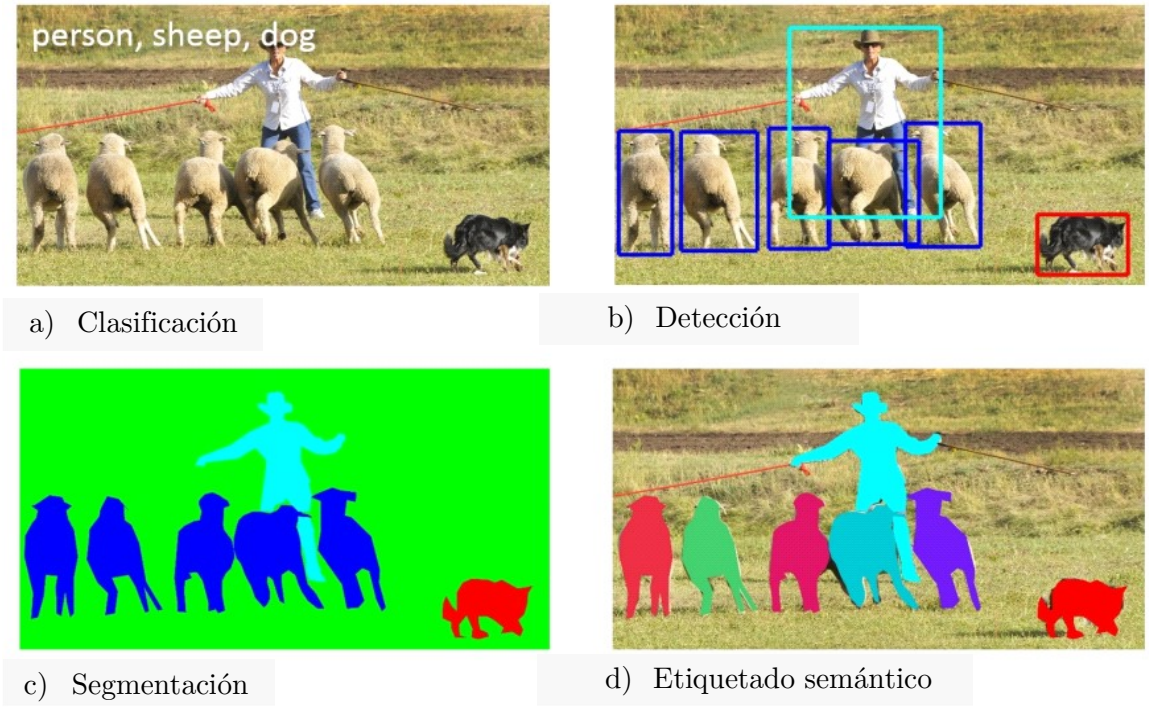

Figura 3.8: Extracción de la base de datos COCO [10]

La base de datos ha sido construida en base a un proceso de anotación: en primer lugar, se realizó un **etiquetado por categorías** de cada una de las imágenes. Seguidamente se **localizó** la situación de cada una de las instancias, procediendo a su **segmentación**, definiendo de forma precisa cada una de las instancias etiquetadas y eliminando falsos positivos. Toda esta tarea ha sido realizada **manualmente** (70.000 horas de trabajo).

Para este proyecto, se utilizará la primera parte de la base de datos, liberada en 2014, la cual contiene la mitad de las imágenes. Estas se encuentran subdivididas en 3 bloques: **80.783 imágenes para entrenamiento**, **40.504 para validación** y **40.775 para test**. Además, se hará uso de las herramientas de cálculo de **métricas** proporcionadas por la propia base datos, detalladas en el **anexo D**, así como la colección de **etiquetas** y el conjunto de **5 *captions*** para cada una de las imágenes.

- A transit bus riding down a street with other traffic.
- A blue bus on a street behind some cars.
- A blue bus is riding behind two other vehicles.
- A bus is driving on the street behind some cars.
- A street view of a moving transit bus.

Figura 3.9: Ejemplo de *caption* de la base de datos COCO [31]

3.3.2 Preprocesado

Tras escoger la base de datos a utilizar para el entrenamiento, se procederá al preprocesado de la información disponible para su correcta utilización a la hora de entrenar la red. Para el **preprocesado, entrenamiento y generación de la red**, se partirá del **repositorio *Self-Critical Sequence Training for Image Captioning*** [1]**,** el cual ha sido estudiado, modificado y adaptado para este proyecto. Esta red se encuentra basada en la utilización de **modelos de atención,** mapeando regiones de la imagen relevantes para la generación de cada palabra mediante el uso de la *sentinel gate*, y **SCST (*Self-critcal Sequence Training*),** aplicando conceptos de aprendizaje de refuerzo para la optimización y ajuste del modelo previamente entrenado.

Se puede dividir el preprocesado en **dos categorías** diferentes. La generación del archivo de vocabulario y captions, y la obtención de las características de las imágenes de la base de datos.

3.3.2.1 Obtención de *captions* y vocabulario

En primer lugar, se **generará el fichero que contenga el vocabulario** y los *captions* de cada una de las imágenes. Este será necesario para el preprocesado de los ficheros finales que realizaran el ajuste al modelo entrenado. Para ello, se tomará el fichero de información proporcionado por la base de datos COCO, del cual se extraerán los *captions* disponibles. A partir de estos *captions*, se construirá el **vocabulario** que contendrá la recopilación de todas las palabras. Tras esto, se procederá al **conteo** de cada una de las

palabras, eliminando del vocabulario aquellas que **no se repitan más de 5 veces** entre todos los *captions*. De esta forma será posible limitar la longitud del vocabulario y su complejidad.

Tras esto, se comprueba cada uno de los *captions*, revisando que la longitud de estos no sobrepase la indicada y se **substituye aquellas palabras que no hayan sido almacenadas en el vocabulario por el *token* UNK(*Uknown)***. Los *captions* procesados y el vocabulario generado, se codifican en un fichero *h5*, almacenando cada *caption* junto con su **longitud**, **comienzo** y **final**. Por último, se genera otro fichero que contenga la **ruta** a cada una de las imágenes, su **identificador** numérico, el ***split*** al que pertenecen (entrenamiento, validación o test) y una copia del **vocabulario**. En este caso, se ha generado un vocabulario que contiene un total de **9487 palabras**, incluyendo el *token* UNK.

3.3.2.2 Obtención de características de la imagen mediante ResNet

En segundo lugar, se realiza un **procesado de las imágenes** de entrenamiento mediante la red ResNet, extrayendo sus características, las cuales serán durante **el entrenamiento**. Para ello, se carga la red **pre-entrenada** [1], se lee el fichero *json* que contiene las imágenes de la base de datos y se generan los ficheros donde se almacenaran las características. A continuación, se realiza un **pre-procesado** de las imágenes, transponiendo las capas de estas, normalizándolas y convirtiéndolas de escala de grises a RGB mediante una triplicación estas. Por último, se **introduce cada imagen en la red, extrayendo sus características** y almacenándolas en los ficheros previamente generados.

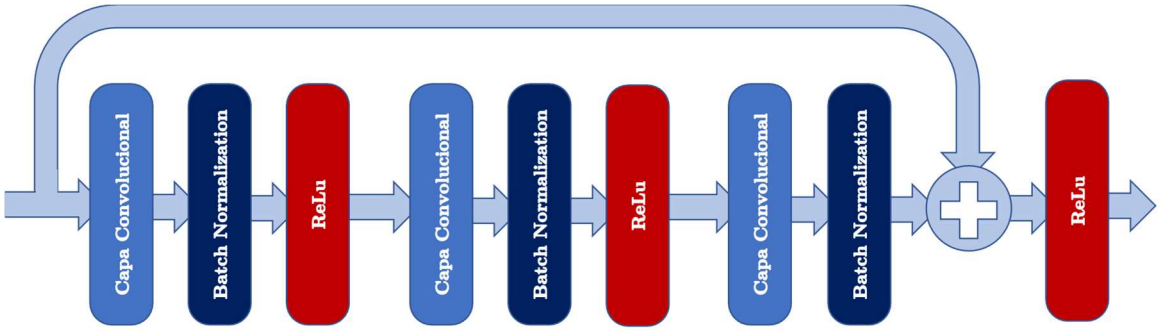

Figura 3.10: Bloque convolucional *Bottleneck*

La red utiliza una arquitectura Resnet101, compuesta por **101 capas convolucionales**. Está generada mediante la agrupación de **bloques *Bottleneck***. Cada uno de estos bloques está compuesto por una combinación de capas **convolucionales**, de **normalización** y capas de activación **ReLu**, como se puede observar en la **figura 3.10.** De esta forma, es posible generar la arquitectura mediante la **repetición de estos bloques**, variando el número según el tamaño de la red ResNet. Para el caso de 101 capas, se utilizará una estructura compuesta por un conjunto inicial de 3 bloques de 64 neuronas de entrada, seguido por 4 bloques de 128 neuronas, 23 bloques de 256 y finalmente 3 bloques de 512 neuronas. Tras estos bloques se encontrará una capa *average pooling* para reducir las dimensiones de las características y una capa *fully-connected* que extraerá el ***scoring* de las 1000 clases**. La arquitectura completa se puede observar en **la figura 3.11**.

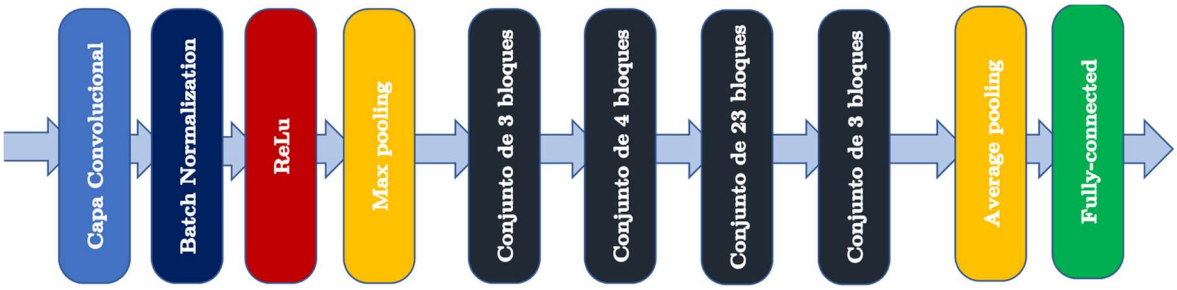

Figura 3.11: Estructura ResNet 101

3.3.3 Entrenamiento y generación del modelo

Una vez obtenidas las características, se procederá al entrenamiento. El código a partir del que se ha partido para la generación del modelo se encuentra en el repositorio *Self-critical Sequence Training for Image Captioning* [1].

La descripción automática (*captioning*) [56] es el proceso por el cual se obtiene una **descripción de la imagen presentada** mediante la combinación de técnicas de procesado de lenguaje natural y vision por computador (*computer visión*). Compuestos por una red neuronal **convolucional** que se encarga del pre-procesado y la extracción de características, tal y como hemos visto en el punto anterior, y de una red neuronal **recursiva** RNN. Esta última se encargará de realizar el modelado del lenguaje a nivel de palabra. En este caso, la red utilizada para la generación de los *captions* está basada en **modelos de atención** [34], **SCST** (*Self-Critical Sequence Training*) [33] y una variación del descenso de gradiente.

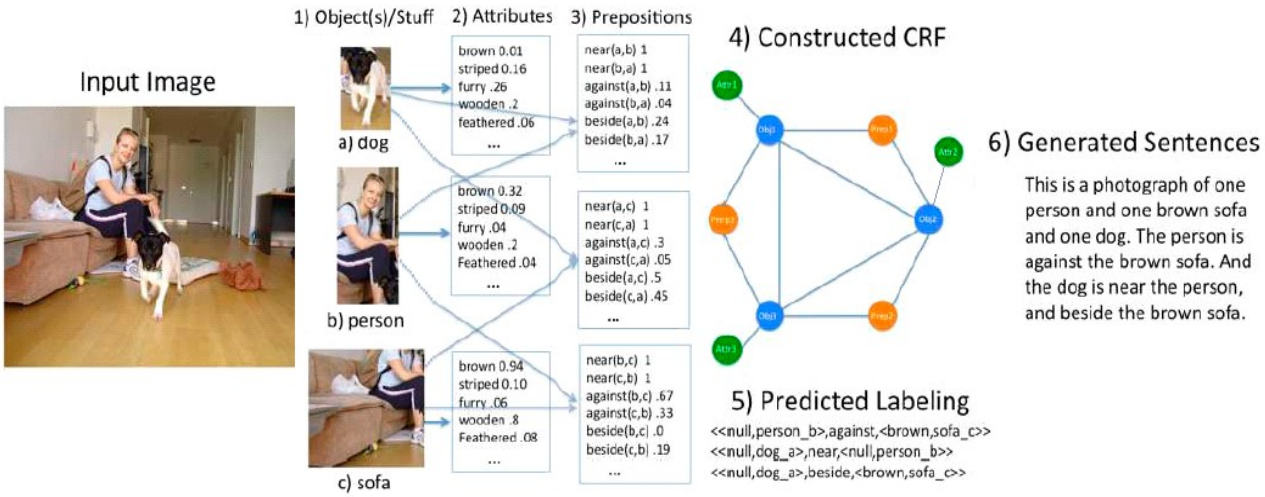

Figura 3.12: Esquema de funcionamiento del *captioning* [32]

Los modelos de atención permiten el mapeado **de regiones de la imagen relevantes para cada palabra generada** de la descripción. De esta forma, en lugar de usar una representación estática agrupada espacialmente de la imagen, los modelos de atención redimensionan las funciones de entrada espaciales de las CNNs para enfocarse en regiones específicas de la imagen.

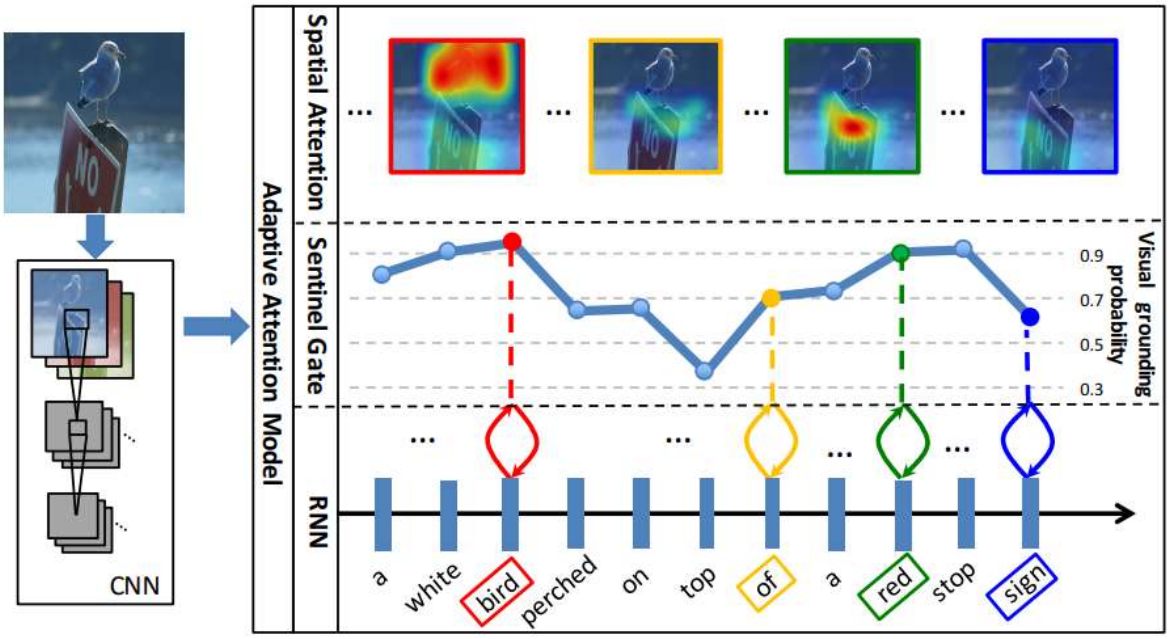

Figura 3.13: Ejemplo del comportamiento del modelo de atención [34]

Sin embargo, como **no todas las palabras tienen correspondencias visuales** (algunas preposiciones, artículos, conjunciones…) o pueden ser generadas mediante la correlación del lenguaje (encima, debajo…), se determinará mediante un nuevo componente si basar la generación de la descripción en la información espacial proporcionada por el modelo de atención o en la propia descripción. Mediante la llamada ***sentinel gate***, es posible **determinar a donde "mirar"** al generar las palabras.

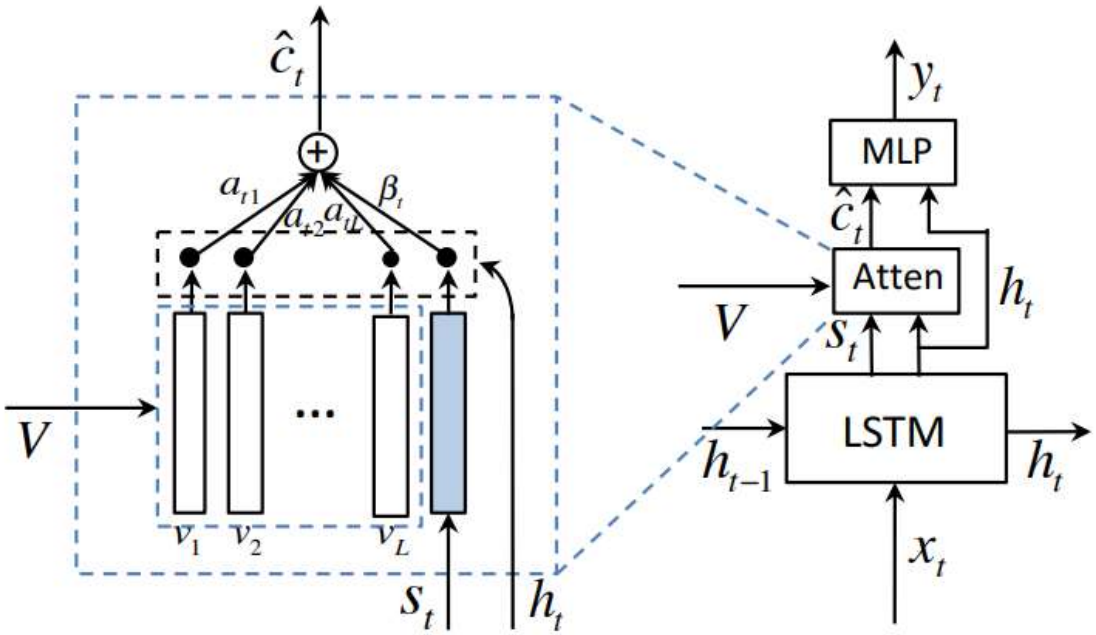

Figura 3.14: Modelo de atención basado en *sentinel gate* [34]

Respecto al algoritmo de actualización de los pesos, se utiliza una versión **evolucionada del descenso por gradiente estocástico, conocida como *Adam*** (*Adaptative Moment Estimation*) [51]. Este método calcula tasas de aprendizaje adaptativas para cada uno de los parámetros a partir de estadísticas de primer y segundo orden de los gradientes. Resultados empíricos han demostrado que *Adam* funciona bien en la práctica y que presenta **una convergencia más rápida** que otros métodos de optimización por gradiente [50] [52].

Por último, tras realizar el entrenamiento, se procederá a **un ajuste mediante SCST**. SCTS posee todas las ventajas de los algoritmos de refuerzo y optimiza la precisión a nivel de secuencia. Además, en la práctica, SCST tiene una **varianza mucho menor** y puede ser entrenado de forma más **eficiente** mediante la utilización de *mini-batches* de muestras. La idea central de SCST es el uso del aprendizaje de refuerzo junto con la recompensa obtenida por el modelo anterior al ajuste.

Los sistemas de *captioning* [32] son tradicionalmente entrenados usando las perdidas por entropía cruzada. Sin embargo, para **optimizar** directamente las métricas NLP y abordar el problema del sesgo de exposición, se puede orientar el modelo al **aprendizaje de refuerzo**. De esta forma, el núcleo LSTM de la red puede verse como un "agente" que interactúa con un "entorno" externo (las palabras y las características de la imagen). Después de cada iteración, el agente actualiza su estado interno (valor de las celdas, pesos y estados ocultos de la LSTM). Al generar el *token* de final de secuencia (EOS) el agente observa la recompensa (por ejemplo, la puntuación de la métrica CIDEr de la descripción generada) calculada al ser comparada con los *captions* de validación de la base de datos y actúa en base al valor de esta, siendo el **objetivo** de la red la **minimización del valor negativo** esperado, maximizando la recompensa obtenida.

3.3.4 Métricas

La generación de *captions* para imágenes es un **problema desafiante** para la inteligencia artificial y el aprendizaje automático (*machine learning*). Esto es debido a la **dificultad de evaluar objetivamente la descripción** de una imagen, generando una gran variación de una persona a otra debido a la subjetividad de la tarea. Por ello, se hará uso cuatro **métricas** diferentes [35], las cuales toman en cuenta diferentes conjuntos de aspectos. De esta forma, es posible obtener una medida de la calidad de las descripciones obtenidas por el modelo, y así poder realizar un proceso de **comparación y ajuste.**

El objetivo final es obtener la calidad de un ***caption* candidato** para una **imagen** dada una colección de *captions* de **referencia**. Estos *captions* se representarán mediante el uso **de n-gramas**, entendidos como un conjunto de una o más palabras ordenadas. Las principales métricas utilizadas son **BLEU** (*Bilingual evaluation understudy*)**, ROUGE** (*Recall-Oriented Understudy for Gisting Evaluation*)**, METEOR** (*Metric for Evaluation of Translation with Explicit Ordering)* **y CIDEr** (*Consensus-based Image Description Evaluation*)**.** Tras analizar las diferentes métricas, se utilizará como **prioritaria CIDEr, debido a su mejor rendimiento con frases individuales** y su mayor variabilidad, pudiendo observar cambios entre modelos que serían casi imperceptibles al ser comparados mediante el resto de las métricas.

Los conceptos en los cuales se basan estas métricas se encuentran **explicados con mayor profundidad** en el **Anexo D**.

3.4 Generación del resumen mediante NLP

El tercer y último bloque del sistema tomará las descripciones de los *frames* principales generadas por el bloque anterior, procesándolas para eliminar aquellas que no aporten una información relevante y **generando el resumen final**. Una vez generadas y extraídas las descripciones en modo texto, se **calcularán los *embeddings*** de cada una de estas mediante el uso de una red BiLSTM basada en la base de datos SNLI (***Stanford Natural Language Inference***) [37] y en el algoritmo GloVe (***Global Vectors for Word Representation***) **[38]**. Mediante estos *embeddings,* se **calcularán las relaciones** entre cada una de las descripciones mediante la **distancia coseno**, para posteriormente eliminar aquellas con una relación demasiado alta o demasiado baja. Por último, se **generará el resultado final** mediante el uso del algoritmo LSA.

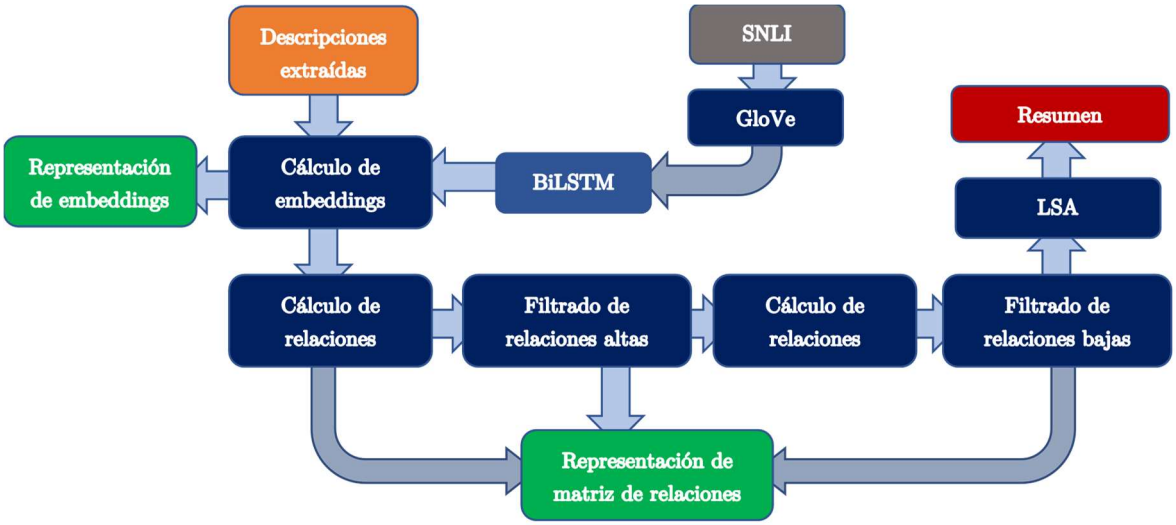

Figura 3.15: Composición del generador de resúmenes

En primer lugar, se realizará el cálculo de los *embeddings* de cada una de las descripciones. Para ello utilizaremos una red neuronal BiLSTM basada en GloVe (*Gloval vectors*) y entrenada a partir de la base de datos SNLI (*Stanford Natural Language Inference*).

3.4.1 SNLI y GloVe

El corpus SNLI [36] [37] es una **colección de 570.000 pares de oraciones** en inglés escritas manualmente para una clasificación equilibrada mediante las etiquetas de implicación, contradicción y neutralidad, que respaldan la tarea NLI (***Natural Language Inference***), también conocida como **reconocimiento de la vinculación textual** (RTE). Su objetivo es servir como punto de referencia para la evaluación de sistemas de representación de texto, especialmente aquellos inducidos por métodos de aprendizaje de representación, así como recurso para el desarrollo de modelos LP de cualquier tipo. SNLI es un campo de pruebas ideal para las teorías de representación semántica.

Hasta su creación, había sido imposible aprovechar este potencial debido a la naturaleza limitada de los recursos existentes para la tarea de **inferencia de lenguaje**

natural. Esta base de datos busca remediar ese problema mediante un nuevo corpus de frases a gran escala. SNLI presenta valiosos datos de entrenamiento y un banco de pruebas desafiante para la aplicación del aprendizaje automático a la representación semántica.

A su vez, GloVe [38] es un **algoritmo de aprendizaje no supervisado** para obtener la representación vectorial de palabras. Es una extensión del método *word2vec* [26], basado en la combinación de las técnicas de factorización matricial según las estadísticas globales como LSA y el aprendizaje basado en contexto local de *word2vec*. En vez de utilizar una ventana para definir el contexto local, GloVe construye una **matriz explicita** de contexto de palabras **usando la estadística** encontrada en el corpus del texto. El resultado es un modelo de aprendizaje que da lugar, en general, a mejores *embeddings*. De esta forma, se utilizará la base de datos SNLI junto con **un conjunto de vectores pre-calculados** mediante el algoritmo GloVe para la generación de los *embeddings* de las descripciones, tal y como se explica en el siguiente apartado.

3.4.2 Cálculo de *Embeddings*

En los apartados anteriores, se ha descrito como obtener una representación vectorial o *embedding* debido a su importancia. Sin embargo, en este caso intentaremos capturar las relaciones de múltiples palabras y frases en un vector único. Mientras que, en problemas relacionados con visión por computador, el uso de redes convolucionales es la opción predominante, la **codificación de frases presenta una gran variabilidad**, pudiéndose encontrar arquitecturas basadas en redes convolucionales, redes recurrentes o incluso esquemas mucho más sencillos según la composición de palabras.

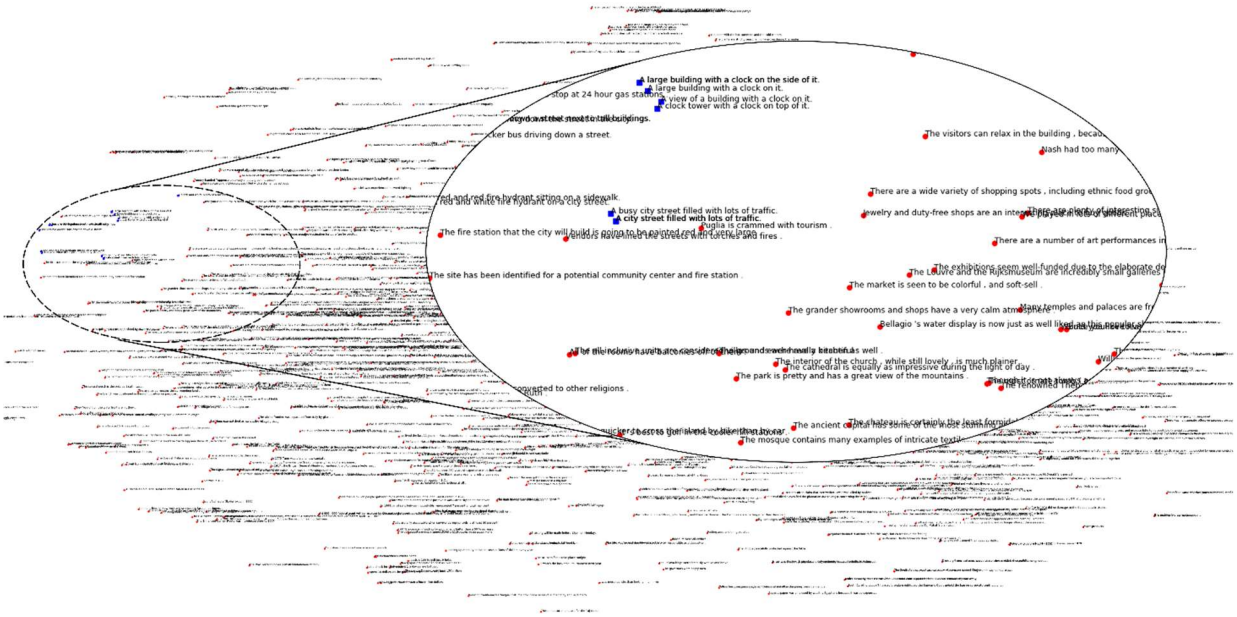

Figura 3.16: Ejemplo de representación de *embeddings* de frases

En este caso, se ha seleccionado una **red neuronal BiLSTM basada en *max-pooling*** [40]**,** la cual utiliza el valor máximo de cada dimensión de las neuronas ocultas para combinar ambas redes LSTM de las que está compuesta y formar un vector de tamaño fijo. Esta red se encuentra ya pre-entrenada mediante el uso de la base de datos SNLI y un

conjunto de vectores calculados mediante el algoritmo GloVe. Se encuentra disponible en el repositorio ***InferSent*** [4].

De esta forma podremos realizar una conversión de cada una de las descripciones a vectores de **dimensión 4096**. Para su representación, se tomará una muestra de los *embeddings* de 10.000 frases y se realizará la **disminución de sus dimensiones** mediante la herramienta t-SNE (*t-distributed Stochastic Neighbor Embedding*) [39]. De esta forma, redimensionando a 2 dimensiones, es posible **visualizar la agrupación de las frases en *clusters* según su contexto**, observando una mayor proximidad en aquellas que comparten o hablan de temas relacionados.

3.4.3 Cálculo de relación entre descripciones

Tras la obtención de los *embeddings* de cada una de las descripciones, se procederá al cálculo de la relación entre estas. Esto permitirá obtener un **indicador de cohesión objetivo** entre dos descripciones, pudiendo realizar un post-procesado a partir de estos.

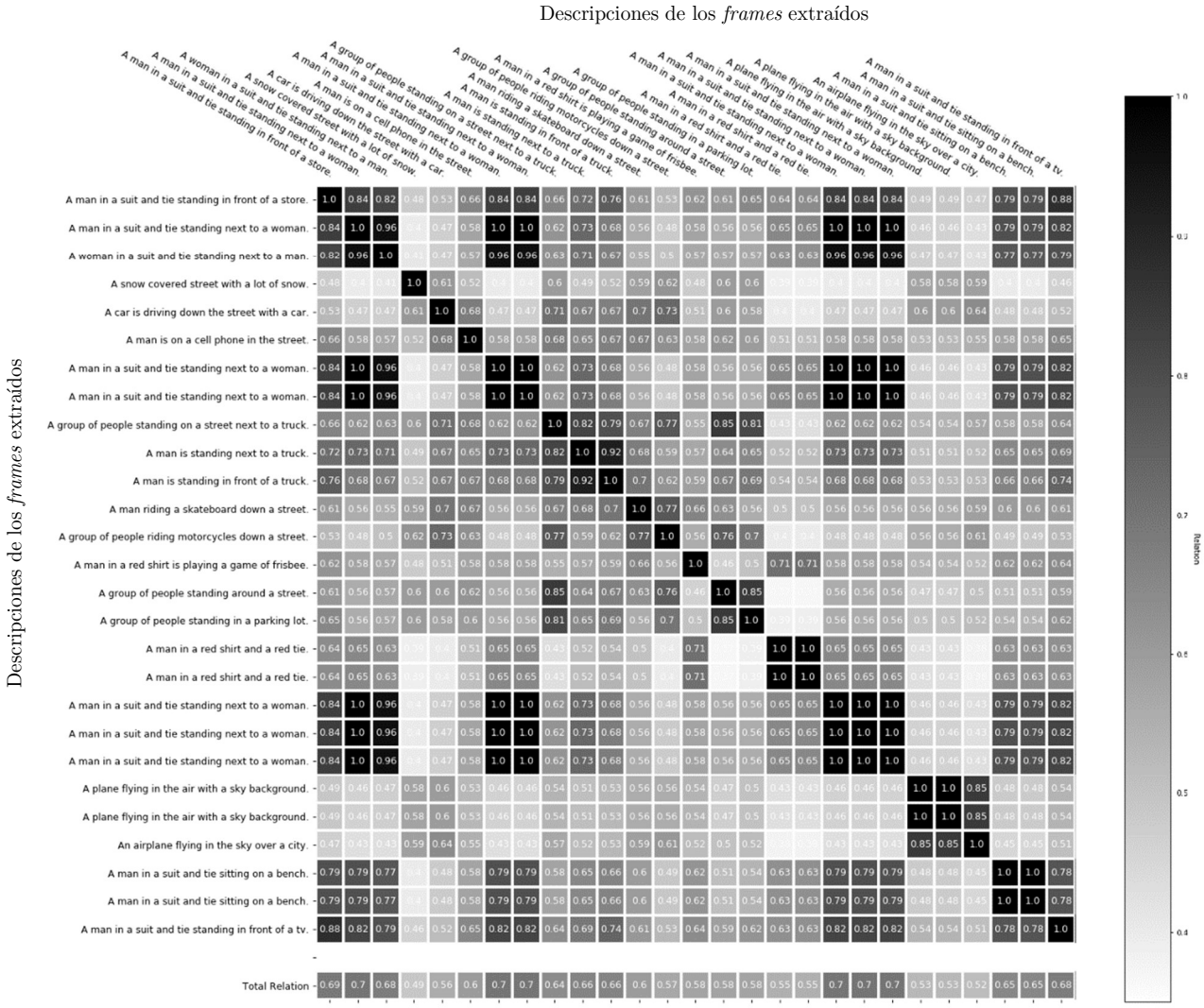

Figura 3.17: Matriz de relación entre descripciones

Para la obtención de esta medida de relación, se utilizará la **distancia coseno** [41]. Esta mide la **similitud existente entre dos vectores en un espacio** a partir del producto escalar con el que evaluar el valor del coseno del ángulo comprendido entre ellos.

$$\textbf{\textit{Similitud}} = cos\boldsymbol{\theta} = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{|\boldsymbol{u}||\boldsymbol{v}|} \tag{3.7}$$

La distancia coseno **no debe ser considerada como una métrica** debido a que no cumple la **desigualdad triangular** (suma de dos lados mayor que el lado restante), pero en este caso permitirá obtener un indicador del nivel de semejanza. A la vez que son calculadas las similitudes entre todos los pares, se calculará también el valor de similitud general de cada una de ellas mediante su media. De esta manera obtendremos la **relación individual** entre cada par y su **relación global**, como se puede observar en la **figura 3.17**.

3.4.4 Filtrado de descripciones según su grado de relación

Una vez obtenidas las distancias entre cada una de las descripciones como medida de la similitud entre estas, se realizará un **filtrado de aquellas que no aporten información** relevante.

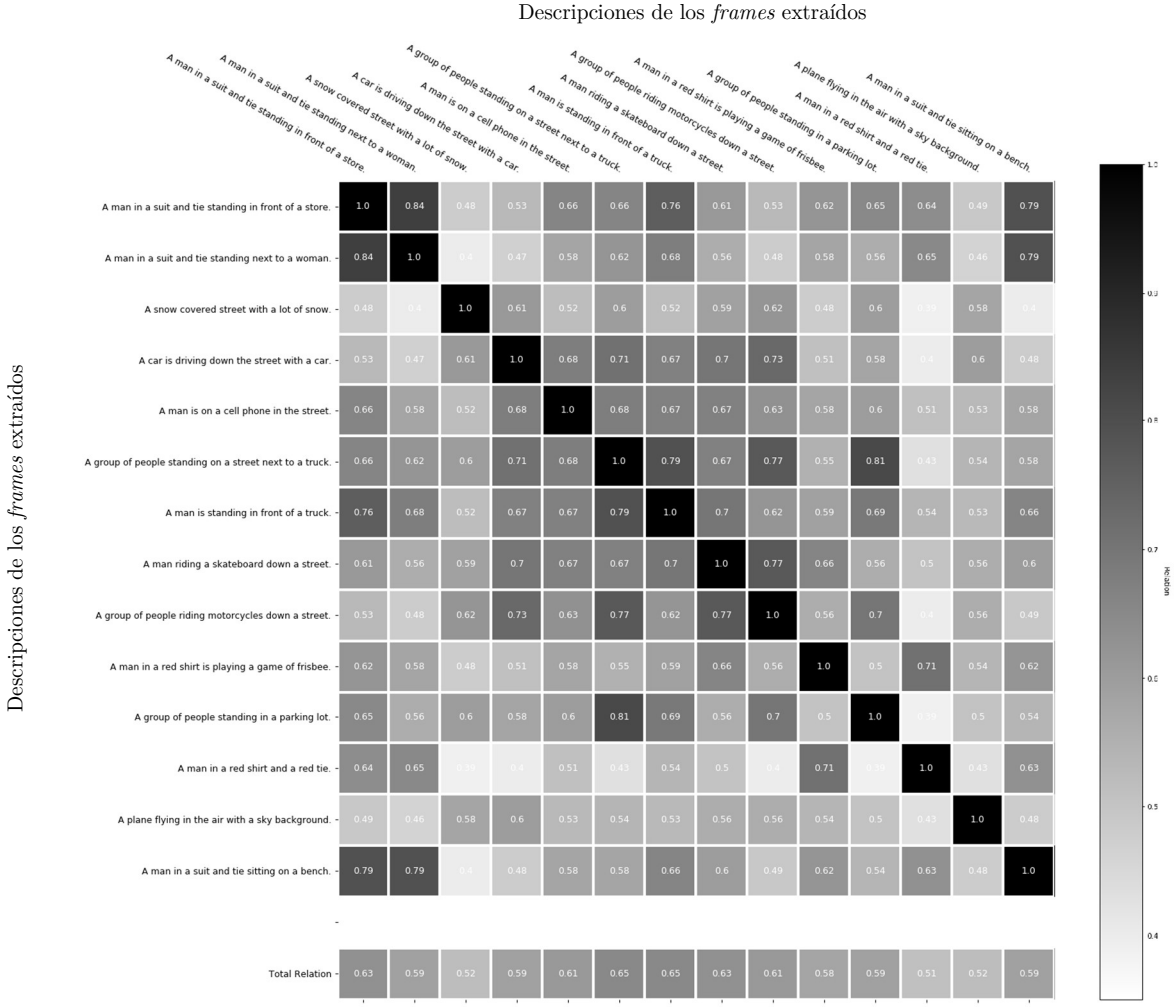

Figura 3.18: Matriz de relación entre descripciones filtradas

Para ello, se comprobarán aquellos pares de **descripciones que tengan un nivel de similitud mayo**r que el del umbral seleccionado. De este par, eliminaremos aquellas que posea un menor nivel de relación global. El umbral seleccionado se encuentra habitualmente **entre 0.85 y 0.90**, ya que se ha observado que la mayor parte de descripciones que presentan un nivel de **relación mayor coinciden en prácticamente la totalidad de su significado**, como podemos observar en la **figura 3.18**. De esta forma se eliminarán las descripciones de *frames* pertenecientes a la misma escena que se hayan generado por error o que sean resultado del **sobremuestreo** de *frame* principal aplicado al final del primer bloque del sistema. Tras esto, se volverán a **calcular** el nivel de similitud de las descripciones restantes.

Una vez recalculados los niveles de semejanza, realizaremos la misma operación que en el paso anterior, eliminando aquellas descripciones con un **grado de similitud general menor que el umbral**, ajustando este a valores comprendidos **entre 0.45 y 0.55**. Se eliminarán las descripciones que no posean relación con el resto, debido a un error o a la representación de conceptos que apenas tienen peso en el total del video. De esta forma se obtendrán las descripciones a partir de las cuales se generará el resumen.

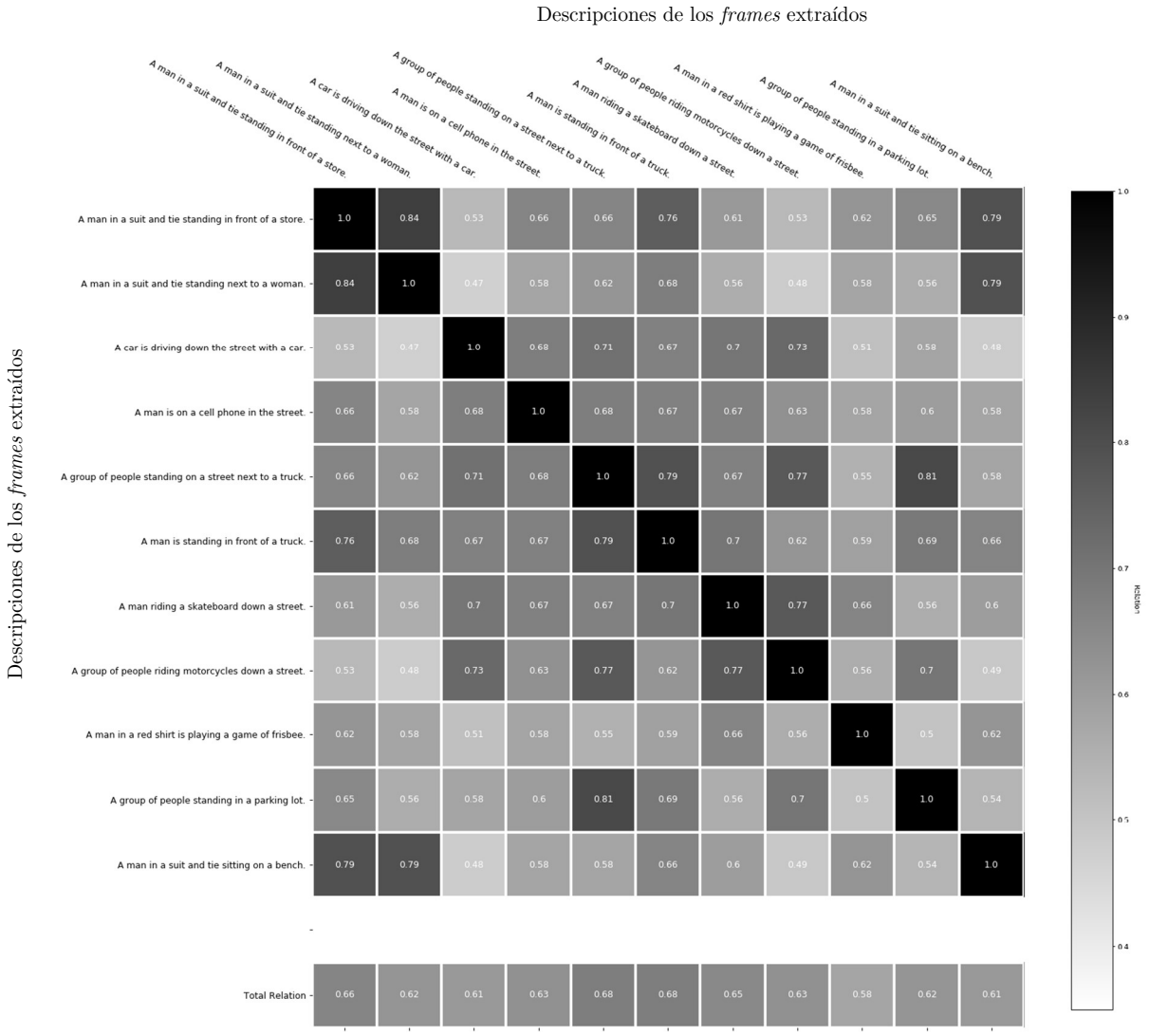

Figura 3.19: Matriz de relación entre descripciones finales

3.4.5 Generación del resumen mediante LSA

Finalmente, una vez obtenidas las descripciones filtradas mediante la distancia entre su representación por *embeddings*, se procederá a la generación del resumen final. El método seleccionado para esto es el **algoritmo de procesado de lenguaje natural LSA** [43]. Permite obtener la relación entre las palabras contenidas en las descripciones y las mismas descripciones, pudiendo obtener una **medida de la importancia** de cada una, seleccionando las $\mathbf{N_r}$ más importantes, las cuales compondrán el resultado final. La aplicación del algoritmo LSA para la generación de resúmenes se puede reducir en **3 pasos principales**: la **generación de la matriz** de entrada a partir un vocabulario, la **descomposición SVD** y la **selección de descripciones**.

En primer lugar, se **compondrá el vocabulario** a partir del cual se generará la matriz de entrada. Para ello se tomarán todas las palabras contenidas en cada una de las descripciones. La forma en la que el vocabulario es generado es muy importante, debido a que **afectará a la forma que tendrá la matriz de entrada** y posteriormente al resultado de las matrices calculadas mediante SVD. El sistema tiene implementados la generación de matrices por frecuencia, representación binaria y *tf-idf*. Estos son los que **mejor compromiso** presenta en complejidad y precisión de los resultados, siendo fácilmente intercambiables por la modificación de uno de los parámetros.

Tras la generación de la matriz de entrada, se aplicará la **descomposición SVD** para la obtención de las matrices $\mathbf{U}$, $\mathbf{\Sigma}$ y $\mathbf{V^T}$ tal y como queda detallado en el **apéndice D** de la memoria. Por último, a partir del resultado de la composición, se procederá a la **selección de las de mayor relevancia** mediante el uso del algoritmo ***cross method.*** Este ha sido seleccionado debido a su eficacia en comparación con los demás. De esta forma serán seleccionadas las $\mathbf{N_r}$ **descripciones de mayor relevancia**, siendo estás las que compondrán el resultado final. Las descripciones pueden **mantener el orden** con el que han sido seleccionadas u **ordenarse según su orden de aparición** en el video. Habitualmente se trabajará con longitudes de resumen ($\mathbf{N_r}$) comprendidas **entre 4 y 6 descripciones**.

Para mayor detalle, los métodos de creación de matrices, el cálculo de la matriz SVD y los algoritmos de selección se encuentran descritos en el A**nexo C** de la memoria.

Capítulo 4

Trabajo experimental y Resultados

En este capítulo se realizará un análisis de los resultados obtenidos en cada una de las etapas del sistema como el resultado final de este.

4.1 Introducción

Para la generación y el análisis de resultados se utilizarán **una selección de videos de corta duración** para agilizar el procesado. Cada uno de estos videos posee unas características diferenciadoras del resto, intentando cubrir el **mayor número de escenarios posibles**.

- **20H:** Segmento del telediario 20H de RTVE (Radio Televisión Española) [59]. Posee cortes de duración media, así como **pantalla dividida** durante la mayor parte de la duración del video. Duración 110 segundos, 18 planos diferentes.
- ***Bunny*:** Video de prueba basado en animación [60]. Posee una mayor resolución, un corto número de cortes y permite ver cómo se comporta el sistema respecto a **videos animados**. Duración 29 segundos, 8 planos diferentes.
- **CA:** Segmento del programa Comando Actualidad de RTVE [59]. Gran número de **cortes y de variabilidad** entre las descripciones. Duración 109 segundos, 22 planos diferentes.
- ***City*:** Plano único del tráfico de una calle de Londres [64]. Permite observar el comportamiento para **un plano único** con gran cantidad de información. Duración 67 segundos, plano único, se considerarán 8 *frames* secundarios relevantes diferentes.
- ***Nature*:** Conjunto de cortes sobre naturaleza y alpinismo [62]. Tiene un gran número de **cambios rápidos**, y gran semejanza entre algunos de ellos. Duración 104 segundos, 30 planos diferentes.
- ***News*:** Segmento del programa de noticias latinoamericano 24 horas [63]. Posee un número estándar de escenas de duración **corta intercaladas con otras de larga duración**. Duración 110 segundos, 16 planos diferentes.
- ***Wildlife*:** Video sobre vida salvaje con un número estándar de cortes [60]. Duración 30 segundos, 7 planos diferentes.

Para la medida del primer bloque se ha realizado un **marcado** de las escenas principales de los videos elegidos. Cabe destacar el problema para realizar una cuantificación objetiva de los resultados finales debido a que **no existen bases de datos ni métricas** con las que comparar los resultados, así como la **subjetividad** del resultado final debido a su **naturaleza semántica**.

4.2 Detector de *frames* principales

El detector utiliza el **muestreo, la diferencia entre los descriptores de los *frames* y su estadística** para obtener los cortes del video y extraer los planos principales. Para ver el rendimiento del sistema y poder ajustarlo correctamente, se realizará un barrido de varios parámetros para observar el comportamiento.

Se usarán **dos indicadores diferentes**, el valor de *recall*, como el número de *frames* detectados en proporción al número de *frames* totales, y el valor de precisión, como el número de *frames* detectados correctamente respecto al total. También utilizaremos el valor-F, como medida ponderada de ambos indicadores:

$$\boldsymbol{Recall = \frac{Detectados}{Detectados + No\ Detectados}} \tag{4.1}$$

$$\boldsymbol{Precision = \frac{Detectados}{Detectados + Falsas\ Alarmas}} \tag{4.2}$$

$$\boldsymbol{Valor - F = 2 \cdot \frac{Recall \cdot Precision}{Recall + Precision}} \tag{4.3}$$

En primer lugar, se analizarán los **valores de muestreo y los factores de diezmado** de los *frames*. Estos parámetros tienen la mayor **implicación en el rendimiento**, por lo tanto, será necesario encontrar un **equilibrio** entre sus valores.

	4	**8**	**12**	**16**	**20**	**30**
20h	44,44	61,11	72,22	77,78	77,78	77,78
news	62,50	56,25	68,75	62,50	68,75	68,75
bunny	37,50	62,50	87,50	87,50	87,50	100
CA	40,91	72,73	77,27	86,36	95,45	81,82
wild	0,00	57,14	57,14	57,14	85,71	85,71
nature	33,33	40,00	50,00	60,00	60,00	60,00
city	50,00	62,50	62,50	75,00	75,00	87,50
Total	**38,38**	**58,89**	**67,91**	**72,33**	**78,60**	**80,22**

Tabla 4.1: % Recall alterando el muestreo en *frames*/s

En la **tabla 4.1** y en la **tabla 4.2** se observa cómo se produce un **incremento para los valores de *recall*** con el aumento del muestreo, y un aumento de la **precisión** para **valores intermedios** debido a la menor extracción de *frames*. Sin embargo, esto **aumenta el tiempo de procesado**, por lo que se tomará un valor intermedio que guarde un compromiso entre eficacia y rendimiento. En la **tabla 4.3** se observa el equilibrio entre precisión y *recall* mediante el valor-F. Se seleccionará un **valor entre 16 y 20 muestras** por segundo. Este **se mantendrá fijo** para el resto de las pruebas presentes en esta memoria.

	4	**8**	**12**	**16**	**20**	**30**
20h	88,89	84,62	86,67	82,35	87,50	93,33
news	90,91	90,00	100	76,92	100	91,67
bunny	100	100	87,50	100	87,50	88,89
CA	100	94,12	89,47	90,48	95,45	90,00
wild	0,00	100	100	100	85,71	75,00
nature	100	100	100	100	100	90,00
city	66,67	71,43	71,43	60,0	66,67	70,00
Total	**78,07**	**91,45**	**90,72**	**87,11**	**88,98**	**85,56**

Tabla 4.2: % Precisión alterando muestreo en *frames*/s

	4	**8**	**12**	**16**	**20**	**30**
20h	59,26	70,97	78,79	80,00	82,35	84,85
news	74,07	69,23	81,48	68,97	81,48	78,57
bunny	54,55	76,92	87,50	93,33	87,50	94,12
CA	58,06	82,05	82,93	88,37	95,45	85,71
wild	-	72,73	72,73	72,73	85,71	80,00
nature	50,00	57,14	66,67	75,00	75,00	72,00
city	57,14	66,67	66,67	66,67	70,59	77,78
Total	**58,85**	**70,82**	**76,68**	**77,87**	**82,58**	**81,86**

Tabla 4.3: Valor-F alterando muestreo en *frames*/s

Modificando el **diezmado** de los *frames* para mejorar el rendimiento del algoritmo, se puede observar como el mejor se obtiene con **valores intermedios**, que permiten un **compromiso entre precisión y *recall***,

	1	**2**	**4**	**8**	**16**
20h	83,33	83,33	77,78	88,89	100
news	68,75	62,50	68,75	68,75	81,25
bunny	100	100	87,50	100	87,50
CA	100	95,45	95,45	86,36	86,36
wild	71,43	85,71	85,71	57,14	100
nature	70,00	60,00	60,00	60,00	73,33
city	75,00	87,50	87,50	100	100
Total	**81,22**	**82,07**	**80,39**	**80,16**	**89,78**

Tabla 4.4: % *Recall* alterando el factor de diezmado

	1	2	4	8	16
20h	100,00	83,33	87,50	88,89	34,62
news	91,67	90,91	100,00	100,00	27,08
bunny	88,89	88,89	87,50	100,00	100,00
CA	95,65	91,30	95,45	90,48	82,61
wild	71,43	75,00	100,00	100,00	41,18
nature	91,30	100,00	100,00	100,00	61,11
city	66,67	70,00	70,00	66,67	21,05
Total	**86,52**	**85,63**	**91,49**	**92,29**	**52,52**

Tabla 4.5: % Precisión alterando el factor de diezmado

Altos valores de diezmado provocan distorsión en la imagen, impidiendo un correcto funcionamiento del detector y obteniendo un **gran número de selecciones, gran parte de ellas falsas alarmas.** Valores **bajos** aumentan mucho **el tiempo de procesado**, sobretodo para videos de alta resolución, **no observándose grandes mejorías** respecto a valores intermedios. Con los resultados obtenidos y según la **tabla 4.6**, se utilizará un **diezmado entre 4 y 8** manteniendo el muestreo a 20 *frames*/s.

	1	2	4	8	16
20h	90,91	83,33	82,35	88,89	51,43
news	78,57	74,07	81,48	81,48	40,63
bunny	94,12	94,12	87,50	100,00	93,33
CA	97,78	93,33	95,45	88,37	84,44
wild	71,43	80,00	92,31	72,73	58,33
nature	79,25	75,00	75,00	75,00	66,67
city	70,59	77,78	77,78	80,00	34,78
Total	**83,23**	**82,52**	**84,55**	**83,78**	**61,37**

Tabla 4.6: Valor-F alterando el factor de diezmado

En las **tablas 4.7 y 4.8** se aprecia la **diferencia de rendimiento** en el selector variando el diezmado y el muestro. El tiempo de procesado disminuye de manera **proporcional al número de muestras** utilizadas, con **diferencias del 86%** entre los valores **16 y 30**, o diferencias del **602% entre el menor y el mayor valor analizado**. Esto se hace mucho **más notable** en el **diezmado**, con una diferencia del **459% entre los *frames* originales y el uso de diezmado 2**, o **un 1581% entre el original y el diezmado 4** escogido tras los análisis. Es importante denotar que la cantidad de tiempo **no disminuye de forma progresiva con el cuadrado del factor de diezmado** como se esperaba, siendo esta disminución menor conforme este aumenta.

	Duración	4	8	12	16	20	30
20h	110	36	70	104	133	186	244
news	110	38	70	110	137	186	255
bunny	29	51	95	150	188	241	344
CA	109	165	311	479	605	774	1132
wild	30	0	22	35	43	60	81
nature	104	41	76	120	148	202	277
city	67	27	50	79	97	134	181
Total	**559**	**358**	**694**	**1077**	**1351**	**1783**	**2514**

Tabla 4.7: Tiempos de procesado en segundos variando muestreo

	Duración	4	8	12	16	20	30
20h	110	2477	449	168	99	82	110
news	110	3708	615	173	70	38	110
bunny	29	4754	817	238	97	49	29
CA	109	9875	1941	765	454	330	109
wild	30	1198	199	55	22	12	30
nature	104	4130	685	189	71	36	104
city	67	2669	443	125	48	25	67
Total	**559**	**28811**	**5149**	**1713**	**861**	**572**	**559**

Tabla 4.8: Tiempos de procesado en segundos variando el diezmado

Tras analizar los parámetros que afectan la calidad del resultado y al rendimiento, se analizaran los que permiten un mejor **ajuste de los resultados sin penalizar el tiempo de procesado necesario**.

	5	10	20	30	40
20h	66,67	77,78	77,78	83,33	77,78
news	75,00	68,75	75,00	81,25	81,25
bunny	100,00	87,50	100,00	100,00	87,50
CA	81,82	81,82	77,27	81,82	72,73
wild	71,43	57,14	57,14	42,86	42,86
nature	46,67	50,00	50,00	50,00	40,00
city	87,50	87,50	100,00	75,00	62,50
Total	**75,58**	**72,93**	**76,74**	**73,47**	**66,37**

Tabla 4.9: % *Recall* alterando valor de anchura de pulso

Se analizarán la **longitud de los pulsos** del convolucionador y el valor de los **umbrales** para el **detector de Histograma de Color** y para el **filtrado por DFT**.

	5	**10**	**20**	**30**	**40**
20h	85,71	87,50	93,33	93,75	100,00
news	85,71	91,67	92,31	100,00	100,00
bunny	88,89	87,50	100,00	100,00	100,00
CA	85,71	90,00	89,47	90,00	94,12
wild	100,00	100,00	100,00	100,00	100,00
nature	93,33	93,75	100,00	100,00	100,00
city	63,64	70,00	80,00	75,00	71,43
Total	**86,14**	**88,63**	**93,59**	**94,11**	**95,08**

Tabla 4.10: % Precisión alterando valor de anchura de pulso

En las **tablas 4.9, 4.10 y 4.11** se puede observar como el aumento de la longitud del pulso provoca una **disminución del *recall*** y **un aumento de precisión** para la gran mayoridad de casos. Se seleccionarán valores comprendidos entre **10 y 20 muestras por ofrecer un buen compromiso** entre *frames* detectados y falsas alarmas.

	5	**10**	**20**	**30**	**40**
20h	75,00	82,35	84,85	88,24	87,50
news	80,00	78,57	82,76	89,66	89,66
bunny	94,12	87,50	100,00	100,00	93,33
CA	83,72	85,71	82,93	85,71	82,05
wild	83,33	72,73	72,73	60,00	60,00
nature	62,22	65,22	66,67	66,67	57,14
city	73,68	77,78	88,89	75,00	66,67
Total	**78,87**	**78,55**	**82,69**	**80,75**	**76,62**

Tabla 4.11: Valor-F alterando valor de anchura de pulso

Se analizarán también **los umbrales de decisión** a partir de la estadística de la señal de diferencias. En las **tablas 4.12 y 4.13** se observa como un **incremento** en el factor que redimensiona el umbral a partir de la varianza implica una **disminución en el valor de *recall* y aumenta la precisión**. Esto es debido a que un aumento del umbral provoca una mayor diferencia para la extracción de los *frames*. Por ello se escogerán **valores comprendidos entre 2 y 3**.

	2		**3**		**4**		**5**	
	Prec	***Rec***	***Prec***	***Rec***	***Prec***	***Rec***	***Prec***	***Rec***
20h	82,35	77,78	86,67	72,22	84,62	61,11	83,33	55,56
news	91,67	68,75	91,67	68,75	100	56,25	100	56,25
bunny	88,89	100	88,89	100	85,71	75,00	100	62,50
CA	86,36	86,36	88,89	72,73	84,62	50,00	90,91	45,45
wild	100	85,71	100	57,14	100	28,57	100	28,57
nature	90,48	63,33	93,75	50,00	93,75	50,00	100	40,00
city	61,54	100	72,73	100	70,00	87,50	85,71	75,00
Total	**85,90**	**83,13**	**91,64**	**74,41**	**88,39**	**58,35**	**94,28**	**51,90**

Tabla 4.12: % Precisión y % *Recall* alterando umbral de decisión

	2	**3**	**4**	**5**
20h	80,00	78,79	70,97	66,67
news	78,57	78,57	72,00	72,00
bunny	94,12	94,12	80,00	76,92
CA	86,36	80,00	62,86	60,61
wild	92,31	72,73	44,44	44,44
nature	74,51	65,22	65,22	57,14
city	76,19	84,21	77,78	80,00
Total	**83,15**	**79,09**	**67,61**	**65,40**

Tabla 4.13: Valor-F alterando umbral de decisión

	0.5		**0.6**		**0.7**		**0.8**	
	Prec	***Rec***	***Prec***	***Rec***	***Prec***	***Rec***	***Prec***	***Rec***
20h	75,00	77,78	81,25	72,22	93,33	61,11	92,86	55,56
news	78,57	68,75	91,67	68,75	100	56,25	100	56,25
bunny	80,00	100	100	100	100	75,00	100	62,50
CA	84,00	86,36	88,00	72,73	86,36	50,00	88,89	45,45
wild	100	85,71	100	57,14	100	28,57	100	28,57
nature	85,71	63,33	90,00	50,00	89,47	50,00	93,33	40,00
city	50,00	100	62,50	100	70,00	87,50	75,00	75,00
Total	**83,88**	**80,32**	**91,82**	**70,14**	**91,31**	**58,35**	**92,87**	**51,90**

Tabla 4.14: % Precisión y % *Recall* alterando umbral DFT

La alteración del **umbral de la DFT** repercutirá en el **valor de precisión** de los videos, ya que bajos valores no consiguen filtrar las falsas alarmas producidas por algún error en los procesos anteriores. También se observa un **filtrado excesivo** para **valores muy altos**, eliminando incluso *frames* que habían sido extraídos de forma correcta.

	0.5	**0.6**	**0.8**	**0.8**
20h	80,00	78,79	70,97	66,67
news	78,57	78,57	72,00	72,00
bunny	94,12	94,12	80,00	76,92
CA	86,36	80,00	62,86	60,61
wild	92,31	72,73	44,44	44,44
nature	74,51	65,22	65,22	57,14
city	76,19	84,21	77,78	80,00
Total	**83,15**	**79,09**	**67,61**	**65,40**

Tabla 4.15: Valor-F alterando umbral DFT

Por ello, se seleccionarán **valores comprendidos entre 0.5 y 0.6** para el cálculo del umbral. Los *frames* detectados debido a falsas alarmas **no repercutirán prácticamente** en los resultados **gracias al filtrado por descripción** que se realizará en el último bloque. Por último, en la **figura 4.1** se muestra a modo de **ejemplo** el resultado obtenido en el video de muestra *wildlife* tras ajustar los parámetros.

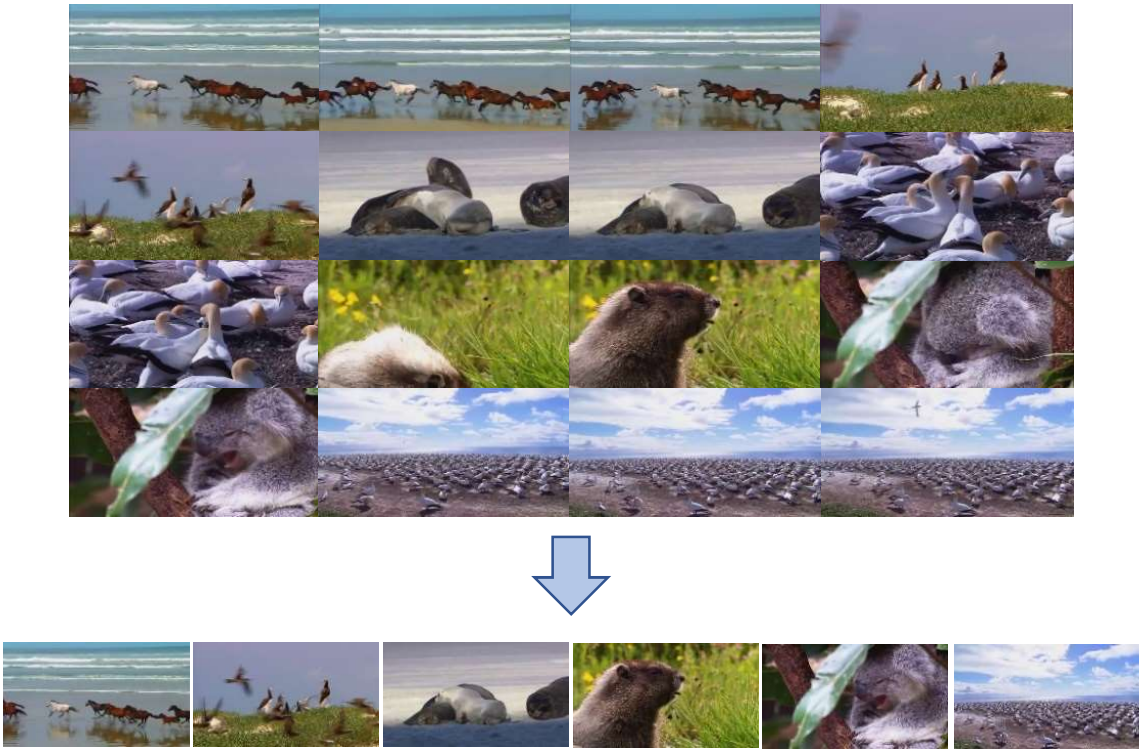

Figura 4.1: Ejemplo de detección de *frames* principales

4.3 Descripción automática de *frames* principales

El descriptor automático de *frames* principales se fundamenta en la **combinación de una red CNN ResNet de 101 capas y una red de *captioning***. La red ResNet se mantendrá constante debido a sus buenas prestaciones, realizando un análisis de la calidad de las descripciones obtenidas mediante **el uso de los *captions* proporcionados por la base de datos COCO** y las métricas descritas en el **apartado 3.3.4** de la memoria.

Se han entrenado y evaluado diferentes versiones de la red de *captioning*. Esta red está **compuesta** por un conjunto de capas, entre las que podemos encontrar capas lineales y de *embeddings*, enfocadas al procesado de las características extraídas por la red ResNet para finalmente generar las descripciones mediante una red LSTM que funciona como núcleo de la red de *captioning* o *encoder*. Esta red LSTM posee una **modificación en la *Input Gate***, en la que sustituye la tangente hiperbólica que genera el vector de valores candidatos por una **operación *max* y añade la *sentinel gate*** desarrollada en los modelos de atención.

Para el entrenamiento de las redes, se ha **partido de los valores proporcionados inicialmente por el repositorio** de origen, realizando diferentes variaciones en estos para intentar conseguir la mayor mejora posible. Entre ellos, se ha seleccionado:

Red CNN de entrada	ResNet de 101 capas
Red RNN *captioning*	LSTM modificada de 1 capa
Tamaño de entrada al *encoder*	512 muestras
Tamaño de las características	2048 muestras
Numero de *epochs*	30
Tamaño del *batch*	10
Probabilidad de *dropout*	50%
Optimizador utilizado	Adam
Ratio de aprendizaje	5e-5
Decaimiento	0.8 cada 3 iteraciones
Alfa	0.9
Beta	0.999
Épsilon	1e-8
Set de validación	5.000 imágenes
Set de entrenamiento	110.000 imágenes

Tabla 4.16: Parámetros utilizados para el entrenamiento

Entre las diferentes versiones del modelo de *captioning*, se realizará la comparación de los dos modelos principales diferentes, **el modelo *FC*, que no incluye el uso de modelos de atención y el modelo *Att2In2*, que, si hace uso de estas mejoras**, presentando un importante incremento de las prestaciones como se ve reflejado en la comparación de sus métricas. Todas estas pruebas han sido evaluadas tanto en su versión **estándar**, obtenida de forma directa tras la finalización del entrenamiento, como la versión **ajustada** mediante el algoritmo *Self-critical Sequence Training*, haciendo uso del concepto de aprendizaje de refuerzo.

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	CIDer	ROUGE-L	METEOR
FC	0.692	0.529	0.386	0.274	0.923	0.498	0.226
FC SC	0.732	0.564	0.422	0.315	0.970	0.537	0.253
FC-2	0.759	0.599	0.457	0.345	1.053	0.557	0.264
FC-2 SC	0.773	0.618	0.482	0.363	1.115	0.568	0.271
Att2In2	0.79	0.621	0.488	0.372	1.265	0.584	0.282
Att2In2 SC	**0.806**	**0653**	**0.505**	**0.385**	**1.278**	**0.587**	**0.285**

Tabla 4.17: Métricas obtenidas de los modelos con COCO

Como se puede observar en la **tabla 4.17**, un **mejor ajuste** de los parámetros de entrenamiento hace **posibles mejoras de un 14%** utilizando el mismo modelo de red (*FC* y *FC2* se diferencian tan solo en algunos valores de entrenamiento).

La red con mejores prestaciones es la que se basa tanto en modelos de atención como en el ajuste mediante SCST. Esta red presenta una **mejora del 38.46%** respecto a la red inicial sin ajustar, y una mejora **del 31.75% respecto a la red ajustada**, comprobando las ventajas presentadas por el uso de los modelos de atención en este sistema. También podemos observar una mejora general de los modelos ajustados mediante SCST respecto a sus versiones estándar de entorno a un **5-6%**, haciendo menos notable esta mejora para los valores más altos obtenidos. Para la realización de estas comparaciones se ha hecho uso de la **métrica *CIDer***, debido a la mayor variabilidad de sus resultados conforme a las modificaciones realizadas en el modelo.

En la **figura 4.2** se puede observar una **comparación entre los cinco *captions* de referencia** disponibles para entrenamiento y validación de la base de datos COCO **y los extraídos** mediante la evaluación de algunos de los modelos entrenados, viendo de forma práctica la mejora en las descripciones de las redes basadas en modelos de atención.

Por último, en la **figura 4.3** se muestra como **ejemplo** la descripción de varios *frames* principales obtenidos de los videos de referencia mediante el detector de *frames* implementado en el bloque anterior. Estos pertenecen a los videos: *wildlife,* un segmento del programa *Comando Actualidad* y *nature*:

Imagen 1

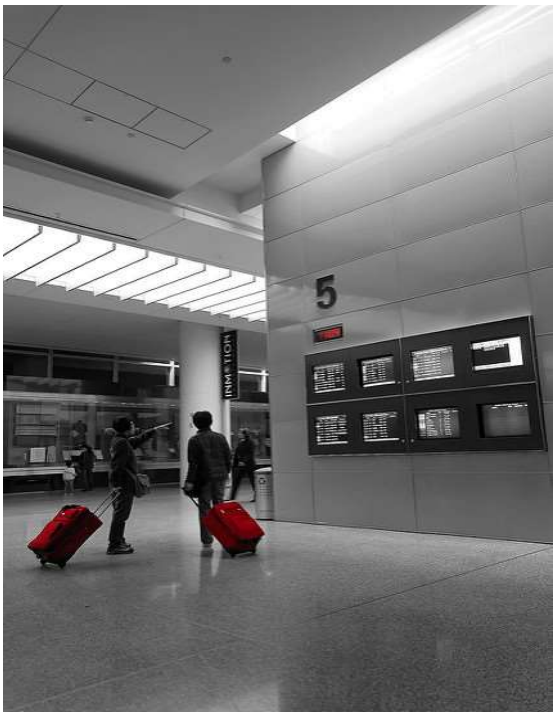

Imagen 2

Imagen 1

- **Ref. 1:** *Bikes are lined up in the store on the floor.*
- **Ref. 2:** *A pink bike in a bike shop with hardwood floors.*
- **Ref. 3:** *A row of bicycles for sale in a store.*
- **Ref. 4:** *A bunch of different bikes on a wooden floor.*
- **Ref. 5:** *A row of bicycles parked next to each other.*

- **Fc:** *A bike is parked next to a wall.*
- **Fc2:** *A bike is parked next to a wall in a room.*
- **Att2in2:** *A group of bikes parked next to a wooden floor.*

Imagen 2

- **Ref. 1:** *Two people in a terminal both pulling red suit cases.*
- **Ref. 2:** *A broad panorama of several people getting together.*
- **Ref. 3:** *A couple of men carrying a couple of red bags of luggage.*
- **Ref. 4:** *Two people holding suitcases while pointing at something.*
- **Ref. 5:** *Man and woman with red luggage in airport.*

- **Fc:** *A man in front of a luggage bag.*
- **Fc2:** *A woman is standing in a room with luggage.*
- **Att2in2:** *A group of people standing in an airport with luggage.*

Figura 4.2: Comparación con los *captions* proporcionados por COCO

A group of horses running on the beach.

A group of chickens standing in a barn.

A woman standing on top of a mountain on a field.

Figura 4.3: Ejemplo de descripción de *frames* principales

4.4 Procesado de *Embeddings* y generación del resumen

El último bloque del sistema se encargará de **generar los *embeddings*** de las descripciones creadas, **filtrar** aquellas que no aporten información relevante y **generar el resumen** final.

Debido a la **naturaleza semántica** de los resultados y a la **falta de bases de datos y métricas** con las que realizar una comparación cuantitativa, el ajuste de los parámetros finales se realizará de manera más subjetiva que en los apartados anteriores. Tras la generación de los correspondientes *embeddings* y el cálculo de sus relaciones, se procederá a ajustar los valores de filtrado.

En primer lugar, se realizará el ajuste de la cota superior, analizando para ello el valor de relación entre las descripciones obtenidas y el contenido de estas. En las **tablas 4.18** y **4.19** se observa la relación entre una de las descripciones seleccionadas como referencia y aquellas con altos valores de relación con esta.

A red double decker bus driving down a street	**Valor de relación**
A red double decker bus driving down a street	1
Two red double decker buses parked on a street	0.91
A white bus is driving down a city street	0.84
A white bus is parked on the side of the street	0.79

Tabla 4.18: Comparación con referencia para alta relación

A flock of birds staying on a beach	**Valor de relación**
A flock of birds staying in the water	0.87
A flock of birds staying in the ground	0.85

Tabla 4.19: Comparación con referencia para alta relación

Se puede observar como de forma general, la mayor cantidad de descripciones que superan **un valor de relación en torno a 0.85** presentan una **similitud muy alta que no aporta información nueva**. De esta manera se puede **filtrar** todas aquellas descripciones que hayan sido **generadas por error debido a falsas alarmas** en el primer bloque o las **obtenidas mediante el sobremuestreo** de *frames* principales, en el caso de que varios *frames* extraídos de una misma escena aporten la misma información. Se repetirá este proceso para la cota inferior, analizando para ello el valor de relación general de cada una de las descripciones.

Descripción	**Valor general**
A chocolate cake sitting on top of a table	0.52
A black dog sitting on top of a table	0.57
A person is sitting in front of a table	0.58
A building with a clock on the side of it	0.61

Tabla 4.20: Comparación para baja relación

En la **tabla 4.20** se encuentran representadas algunas de las descripciones con **menor valor de relación general** obtenidas, extraídas del video de muestra *city*. La elección de la cota inferior debe hacerse de forma **ajustada**, ya que en este caso las **tres primeras** de menor valor no se corresponden con el video debido a **errores** en la extracción de los *frames* o en la descripción de estos, mientras que la **última si es correcta** y puede

aportar relevancia a la generación del resumen. Por este motivo se elegirán **valores que oscilen entorno 0.55 y 0.65** para intentar filtrar de la forma más **eficaz** posible **sin eliminar información útil**.

Por último, se analizarán las diferencias obtenidas en el resultado al **variar la matriz de repetición generada en el algoritmo LSA**. La creación de esta matriz tiene grandes implicaciones en el cálculo de la importancia de las descripciones, por lo que se compararan los rendimientos obtenidos para una matriz **binaria**, una de **frecuencias**, y una generada mediante el **algoritmo *tf-idf***.

Matriz utilizada	***Segmento de CA***
Binaria	*A man standing in front of a street with a tie. A bunch of ducks are standing in the grass. A bird is standing on top of a cage.*
Frecuencias	*A man standing in front of a street with a tie. A bunch of ducks are standing in the grass. A bird is standing on top of a cage.*
TF-IDF	*A dog standing in the grass with a field. A woman is standing in front of a building. A bunch of ducks are standing in the grass.*

Tabla 4.21: Comparación de resúmenes según matriz utilizada

En la **tabla 4.21** se encuentran algunos **ejemplos** de los resúmenes obtenidos para el mismo video, **variando únicamente la matriz** utilizada. Los resultados obtenidos en el caso **binario** y por **frecuencias** son los **mismos** debido a la **corta duració**n del video, provocando el resultado de la **SVD** de ambas matrices sea casi **idéntico**.

Analizando las diferencias junto con el video, se puede llegar a la conclusión de que el uso de ***tf-idf*** genera los **mejores resultados.** Este no solo tiene en cuenta la **aparición de los términos**, sino que también aplica una **penalización** si estos resultan demasiado **comunes**, provocando una **menor cantidad de información redundante**.

4.5 Resultados finales

Tras **analizar los resultados** de cada uno de los bloques para las configuraciones escogidas y **ajustar** los diferentes parámetros del sistema manteniendo un **compromiso** entre el tiempo de procesado y los resultados obtenidos, se analizarán algunos de los aspectos generales y sus consecuencias.

En primer lugar, se realizará una **comparativa del tiempo de procesado** usado **por cada uno de los bloques del sistema respecto al tiempo total** de algunos de los videos seleccionados.

	Etapa	*wildlife*		*news*		*nature*		*media*
		s	*%*	*s*	*%*	*s*	*%*	*%*
1	**Detector**	132,14	58,16	404,32	67,76	187,80	60,88	**62,26**
2	**Descriptor**	12,23	5,38	13,78	2,31	7,53	2,44	**3,38**
3	**Embeddings**	82,83	36,45	178,56	29,92	113,14	36,68	**34,35**
	Resumen	0,01	0,01	0,06	0,01	0,01	0,00	**0,01**

Tabla 4.22: Tiempos de procesado de cada bloque

En la **tabla 4.22** se puede observar como el **primer y el último bloque del sistema concentran** de manera mayoritaria el tiempo de procesado. Esto hace que la selección de los parámetros **de muestreo y redimensionado** del detector afecten en gran medida al tiempo total. También se puede destacar el **elevado coste del cálculo de los *embeddings*** de la relación entre estos de cada una de las descripciones en comparación con el resto de algoritmos.

Por esta razón, es de vital importancia intentar **reducir en la mayor medida posible el número de *frames* erróneos detectados por falsas alarmas** o el valor de sobremuestreo utilizado al final del detector, para no aumentar innecesariamente el tiempo de procesado del último bloque. Por último, mencionar la **rapidez de procesado de la red de neuronal** encargada del proceso de *captioning* en comparación con los tiempos de entrenamiento necesarios para su modelado (llegando a varios días dependiendo de los parámetros utilizados).

Es de importancia destacar la **imposibilidad del sistema para realizar el procesado de los resúmenes en tiempo real**, debido a que algunas de las metodologías empleadas requieren de toda la información para funcionar correctamente. En el primer bloque es necesario poseer todos los *frames* muestreados para **calcular de forma precisa los umbrales de decisión** a partir de la estadística de la señal de diferencias. El segundo bloque si es capaz de realizar el procesado en tiempo real una vez que la red haya sido entrenada. El último, la **red BiLSTM** necesita tener las **descripciones completas** por la forma en la que está estructurada, de igual manera que el **filtrado previo** a la generación del resumen.

Teniendo en cuenta la **muestra de videos presentada en la introducción** de este capítulo y observando los resultados obtenidos, se pueden extraer algunos de los patrones observados en el comportamiento.

- **20H:** Tanto el detector como el descriptor son capaces de realizar de forma correcta su trabajo. Sin embargo, el uso de pantalla dividida en la mayor parte del video hace que **casi todas las descripciones obtenidas hagan referencia a la reportera**, lo que provoca una gran cantidad **de información redundante en el resumen**. Además de aumentar la dificultad de una correcta descripción de la imagen correspondiente a la noticia debido a su menor resolución.

- ***Bunny*:** Selecciona los planos de forma correcta, pero el descriptor tiene **problemas con algunas de las entidades animadas** al **no ser capaz de identificarlas** de forma concreta con un elemento de la base de datos. Por ello oscila entre diferentes tipos de animales o personas.
- **CA:** Se puede observar algunos **fallos del descriptor a la hora de identificar ciertas entidades**, así como **inconexión entre las frases del resumen** generado. Esto es debido a la gran cantidad de escenas presentes y la variabilidad del contenido entre estas.
- ***City*:** Presenta un **buen comportamiento** en general, eliminando satisfactoriamente las falsas alarmas generadas por estar grabado en un plano único.
- ***Nature*:** Se observa cierta **perdida de planos** en el detector debido a **la corta duración de estos y el gran parecido** que presentan entre ellos. También se pueden observar **pequeños fallos en el descriptor** a la hora de detectar ciertas entidades.
- ***News*:** El detector presenta un buen funcionamiento, pero la **redundancia de información** en varios planos y la **complejidad en su descripción** provocan algunos errores en el resultado.
- ***Wildlife*:** El sistema no tiene problema al detectar los *frames* y describirlos, pero si **genera cierta redundancia a la hora de crear el resumen** centrándose en planos muy parecidos.

Por último, en las **tablas 4.23, 4.24, 4.25 y 4.26** se muestran algunos de los **resúmenes obtenidos** para los videos de muestra, en concreto se han seleccionado los resúmenes generados para los videos *city, CA, nature* y *wildlife.*

city
A building with a clock on the side of it. A group of people standing in front of a building. A group of people walking down a street. A red double decker bus driving down a city street.

Tabla 4.23: Resumen generado a partir del video *city*

Segmento de Comando Actualidad
A black cow laying on top of a field. A dog standing in the grass with a field. A bunch of ducks are standing in the grass. A person is standing in the hay. A bird is standing on top of a cage. A man and a woman standing next to a fire hydrant.

Tabla 4.24: Resumen generado a partir del video *CA*

nature
A man sitting on a bench in a field. A man is sitting in the back of a car. A couple of people walking down a snow covered slope. A person standing on a field with a field

Tabla 4.25: Resumen generado a partir del video *nature*

wildlife
A group of horses standing on the beach. A group of birds flying in the grass. A dog laying on top of a beach. A bird is sitting on top of a beach.

Tabla 4.26: Resumen generado a partir del video *wildlife*

Capítulo 5

Conclusiones y Líneas futuras

En este capítulo se valora en conjunto el trabajo realizado y los resultados obtenidos para poder obtener las conclusiones del proyecto, así como las posibles mejoras que podrían desarrollarse en algunas de las partes en un futuro.

5.1 Conclusiones

La gran cantidad de herramientas de búsqueda de documentos multimedia en bases de datos y la **dificultad de la indexación** de este tipo de contenido, hacen necesaria la existencia de sistemas que permitan agilizar el procesado de gran cantidad de datos para su correcto almacenamiento y uso. El **sistema de descripción automática de video propuesto en este TFM facilita** esta tarea. Parte de sistemas ya existentes basados en indexación descriptiva y añade una capa más que permita combinar la información obtenida de los *frames* gracias a los descriptores utilizados. De esta forma se atribuirá al documento datos de **mayor nivel sobre la significación**, realizando así una **combinación entre la indexación semántica y descriptiva**.

Para llevar a cabo este objetivo, se construyó una **arquitectura basada en tres bloques diferenciados**, encargados de la detección de los *frames* principales, su descripción, y la generación de resumen. El detector, basado en el **uso de descriptores** (Histograma de Color, HOG), tras un ajuste de todos sus parámetros, obtuvo unos valores de ***recall* del 85.9**% y **83.13% de precisión** para la selección de videos analizados. A continuación, se implementó un **sistema de *captioning*** basado en la **combinación** de redes neuronales **convolucionales** para la extracción de las características de la imagen y redes neuronales **recurrentes** para la generación de los *captions*. Tras analizar los diferentes modelos disponibles y ajustar sus parámetros de entrenamiento, se obtuvo un **valor 1.278 en la métrica CIDEr** mediante una red basada **en modelos de atención** y **ajustada** mediante **SCST**, mejorando un **38.76%** respecto a las redes más básicas. Cabe destacar la **complejidad** del proceso de *captioning*, siendo este uno de los **puntos clave** del proyecto, y el cual se encuentra en constante desarrollo y actualización.

Las descripciones generadas son **filtradas** según el nivel de **relación** entre estas usando la distancia entre sus *embeddings*, generando finalmente el **resumen** mediante el algoritmo **LSA**. Se puede observar que LSA no se comporta de manera óptima en videos cortos, debido a que el **conjunto de descripciones que estos generan no es lo suficientemente amplio**, **mejorando para videos de mayor** duración. También cabe destacar la **falta de conexión** observada entre las frases del resumen, ya que solo se usa el contenido descriptivo de los *frames* para su generación, así como la dificultad a la hora de realizar una medición de error de los resultados finales. Esto es debido a la **naturaleza**

semántica de estos y a la **falta de una base de datos y unas métricas** que permitan cuantificar los resultados. Por último, hay que destacar la gran variabilidad observada durante el ajuste, lo que provoca que **no exista un conjunto de parámetros** que permita obtener el mejor rendimiento de forma general, dependiendo estos del video introducido y de las **características** presentes de este.

Por tanto, el **objetivo** que se pretendía alcanzar con la realización de este Trabajo Fin de Máster ha sido **completamente satisfecho**. Obteniendo un sistema capaz de generar **resúmenes descriptivos** de forma completamente **automática** para un video dado, **combinando** para ello diferentes técnicas basadas en el **procesado de imagen**, **redes neuronales** y **procesado de lenguaje natural**.

5.2 Líneas Futuras

A la vista de los resultados obtenidos del sistema de descripción automático de video, cabe la posibilidad de aprovechar este TFM como base a nuevos proyectos a través de diversas **líneas de mejora**. Debido a la **naturaleza modular del proyecto**, estando este dividido en **tres bloques** bien diferenciados, es posible encontrar diversas **mejoras para cada uno** de los bloques de forma individual, además de las de ámbito **general**.

En primer lugar, es posible realizar una mejora en la implementación del detector de *frames*, añadiendo las funciones y parámetros necesarios para **extraer de forma mucho más precisa cambios de escena graduales**. También existe la posibilidad de implementar el **detector mediante una red neuronal** convolucional, acelerando en gran medida la velocidad de procesamiento y mejorando los valores de *recall* y precisión conseguidos. Sin embargo, para llevar a cabo este propósito también **será necesaria la implementación de una base de datos de video etiquetada** lo suficientemente extensa para poder entrenar la red. Respecto al bloque de descripción, es posible realizar una **mejora en el modelo** de *captioning* mediante el uso de **bases de datos de mayor complejidad y extensión**, que permitan una mayor identificación de elementos, reduciendo gran parte de los errores observados tras el análisis. Remarcar el aprovechamiento de los **subtítulos** generados por los dos primeros bloques del sistema, los cuales contienen la **descripción de cada uno de los planos extraídos con sus correspondientes marcas de tiempo** para ser visualizados de forma paralela al video.

Como mejora al generador de resúmenes, sustituir el **algoritmo LSA por PLSA** (*Probabilistic Latent Semantic Analysis*), buscando una **mejora** en la calidad y eliminando la **incertidumbre** generada por los valores de **longitud negativos** proporcionados por LSA, al s**ustituir estas longitudes por probabilidades**. Dado la falta de contexto observada entre las diferentes descripciones, añadir una metodología que permita introducirlo, **eliminando la sensación de falta de interconexión entre las frases** que componen el resumen, aumentado la calidad de estos. Para ello, incluyendo un sistema de **transcripción audio a texto**, es posible combinar la información obtenida mediante la descripción de la escena con el contenido contextual de esta.

Por último y de manera general, la **generación de una base de datos** y unas **métricas** que permitan la cuantificación del error generado, permitiendo así una **valoración** mucho más sencilla y objetiva del sistema.

Anexo A

Descriptores Estadísticos

En este anexo se muestra con mayor detalle el cálculo de los descriptores estadísticos utilizados para la detección de los cambios de escena.

A.1 Detección de cambios de escena

La detección de **cambios de escena** saca partido de la extracción de las **características de bajo nivel** de la imagen, como los píxeles, las estadísticas de sub-bloque, bordes… Durante este proceso de detección, se realizará la extracción de las diferentes características visuales para medir el grado de similitud entre distintos *frames*. Esta medida, denominada como **g(n, n + k),** está relacionada con la diferencia o discontinuidad entre un *frame* **n** y un *frame* **n + k**, donde **k≥1**. Existen diferentes métodos para la generación de este valor de discontinuidad en una secuencia de video, siendo uno de los más simples la diferencia absoluta entre dos *frames*:

$$g(n, n+k) = \sum_{x,y} |I_n(x,y) - I_{n+k}(x,y)| \tag{A.1}$$

donde **I(x, y)** es el nivel de intensidad de imagen para el descriptor elegido en la posición **x** e **y**. Generalmente, los métodos basados en diferencias absolutas comparan este valor con un **umbral** para decidir si ha ocurrido o no un cambio de escena. Sin embargo, esta medida de discontinuidad es muy sensible a cambios de luminancia y al movimiento de la cámara, siendo mucho más eficaz el estudio de otras características, como la **estadística del color en la imagen** o la del **movimiento mediante el cálculo de sus gradientes orientados**.

A.2 Detección mediante Histograma de Color

Los histogramas de luminancia consideran solo los cambios y su distribución en la imagen, por lo que, al estar trabajando con imágenes a color, la mejor forma de aprovechar la información disponible es la utilización de su estadística de color. Como alternativa al **espacio de color habitual RGB** [29], podemos evaluar el histograma en el **espacio HSV** (*Hue-Saturation-Value*) debido a su semejanza y su facultad perceptiva. Este espacio está definido según la percepción del ojo humano. Los colores similarmente perceptibles se encuentran situados en niveles cercanos de cuantización, de forma que aquellos que no se parezcan pertenecen a niveles más lejanos. Además, la semejanza entre dos colores puede ser evaluada mediante la distancia en el espacio HSV. Un color puede ser determinado fácilmente por el usuario escogiendo los valores de matiz, saturación y valor.

El único problema reside en la **no existencia de una matriz de conversión directa** entre los espacios **RGB/HSV**, siendo necesaria una conversión del espacio mediante un algoritmo, ya que las imágenes fuente se encuentran expresadas en el espacio RGB, añadiendo una **mayor carga de computación** a la comparación. El uso de HSV también permite una mejor distribución de los bits, ya que el ojo humano es mucho más sensible al matiz que al resto de componentes, siendo posible una mayor asignación de bits a este valor en comparación al resto. Tras la implementación y comparación de ambos tipos de histograma, RGB y HSV, se ha decidido la **utilización de RGB** debido a la similitud de sus resultados y a la necesidad de una conversión adicional en el caso de HSV, lo que añade un tiempo de computación extra al sistema.

Una vez elegido el espacio de color en el que se trabajará, se procederá al cálculo del histograma de los diferentes *frames*. Este método se basa en la **comparación de la medida de discontinuidad** entre *frames* [30]. Este valor será calculado mediante la **medida *Chi-cuadrado*** entre dos histogramas:

$$\boldsymbol{d_{RGB}(X,Y) = \sum_{i=1}^{M} \frac{\left(h_x(i) - h_y(i)\right)^2}{h_x(i) + h_y(i)}} \qquad \textbf{(A.2)}$$

donde $\mathbf{h_x}$ es el histograma de color de la imagen **X** la cual contiene **M** valores de color diferentes. La detección de cambio de escena se calcula mediante la diferencia entre los histogramas de color de *frames* adyacentes pertenecientes a una misma secuencia de video. Esta diferencia es calculada como:

$$\boldsymbol{HistDif[i] = \sum_{j=1}^{M} \frac{(h_i(j) - h_{i-1}(j))^2}{h_i(j) + h_{i-1}(j)}} \qquad \textbf{(A.3)}$$

donde $\mathbf{h_i}$ es el histograma de color con **M** colores en el *frame* **i**, correspondientes a la secuencia de video. En la **figura A.1** se observa el resultado de realizar la diferencia entre histogramas de color para una secuencia de video. Se pueden observar picos cuando grandes discontinuidades entre histogramas ocurren, marcando de esta manera las transiciones abruptas o cortes en el video. Estos son fácilmente reconocibles de otros posibles efectos debido a su gran amplitud. Idealmente, un cambio abrupto **puede ser representado como una función delta**:

$$\boldsymbol{HistDif_{cut}[i] = \alpha_i \cdot \delta(i - i_{cut})} \qquad \textbf{(A.4)}$$

donde $\boldsymbol{\alpha_i}$ representa la amplitud de la función delta y $\mathbf{i_{cut}}$ el número de *frame* en el que ocurre el corte. Aunque no vayan a ser considerados en este proyecto debido a su mayor dificultad de detección, también es posible realizar un modelado de los cambios graduales, teniendo estos la **apariencia de una función rectangular** de menor amplitud que el cambio abrupto, aunque esta amplitud no suele mantenerse constante durante todo el efecto:

$$\boldsymbol{HistDif_{fade\ and\ disolve}[i] = \beta_i \cdot rect\left(\frac{i - i_{fade\ and\ disolve}}{T_{fade\ and\ disolve}}\right)} \qquad \textbf{(A.5)}$$

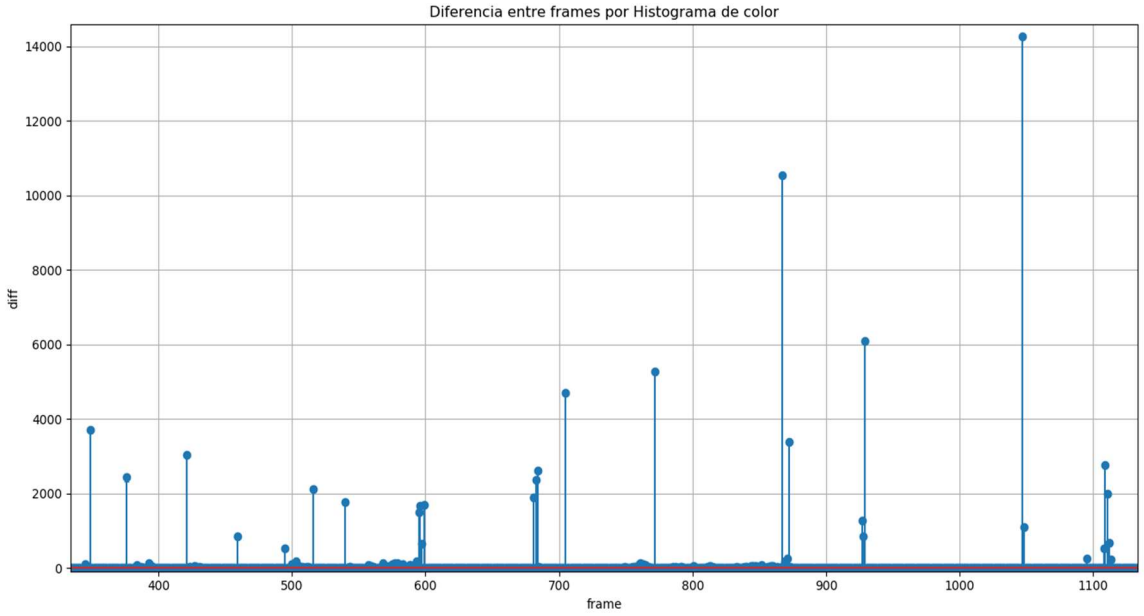

Figura A.1: Resultados de la diferencia entre histogramas de color

Finalmente, la presencia de **objetos y el movimiento** de la cámara producirá señales cuya forma es **muy parecida a los desvanecimiento y cortes graduales**, siendo esta la razón de su mayor complejidad en detección, debido a la necesidad de un análisis exhaustivo para la eliminación de las posibles falsas alarmas.

A.2 Detección mediante Histograma de Gradientes Orientados (HOG)

Además de los cambios basados en el color, también existe la posibilidad de detectar cambios de escena mediante la utilización de **detección de objetos**. La apariencia y forma de un objeto local puede ser descrita por el gradiente del borde, la distribución de densidad de la dirección. Para ello, es posible la comparación de fotogramas adyacentes mediante los **histogramas de gradiente orientado** [17].

Para el cálculo de los gradientes, en primer lugar, se realiza una conversión de la imagen a **escala de grises** y una corrección gamma de 0.75 [53] para reducir el impacto de los cambios en las sombras y luces locales de la imagen, suprimiendo a su vez la interferencia generada por el ruido. De esta forma la imagen se encontrará lista para el cálculo de los gradientes tanto **horizontales como verticales**. Para ello se pueden utilizar un **operador de *Sobel*** [53] con tamaño de **kernel 1**. Una vez calculados los gradientes se procede a su conversión en magnitud y dirección:

$$\boldsymbol{g} = \sqrt{\boldsymbol{g_x^2} + \boldsymbol{g_y^2}} \tag{A.6}$$

$$\boldsymbol{\theta} = \arctan\frac{\boldsymbol{g_y}}{\boldsymbol{g_x}} \tag{A.7}$$

La magnitud de los gradientes se dispara cuando hay un **cambio abrupto** de intensidad, y es nula cuando la región es uniforme. Para cada uno de los pixeles, el gradiente posee una magnitud y una dirección. La imagen basada en gradientes elimina gran cantidad de información innecesaria y **destaca la de mayor importancia**. En otras palabras, aún es posible reconocer aquello destacable en una imagen compuesta únicamente por gradientes.

Tras obtener la dirección y la magnitud, se realizará una **subdivisión de la imagen en sub-bloques** de igual dimensión, para los que se calcularan los histogramas de gradiente. Una de las razones por las que usar un descriptor para identificar las características de una imagen es para describir la información de esta de una forma más **compacta**. Un bloque de 8×8 píxeles contiene un total de 128 valores, contado 64 valores de magnitud y 64 valores de dirección por cada uno de los píxeles contenidos en el bloque. Gracias a la representación en base al histograma, es posible la **representación completa del bloque mediante un histograma de 9 valores** que puede ser almacenado en un array de 9 elementos. No solo la representación es más compacta, calcular un histograma sobre bloques permite una representación más robusta al ruido que en el caso de realizarla de forma individual. La elección del tamaño del bloque va relacionada con el **tamaño de la imagen a procesar y de los elementos a detectar**, siendo elecciones frecuentes 8×8, 16×16 o 32×32 píxeles.

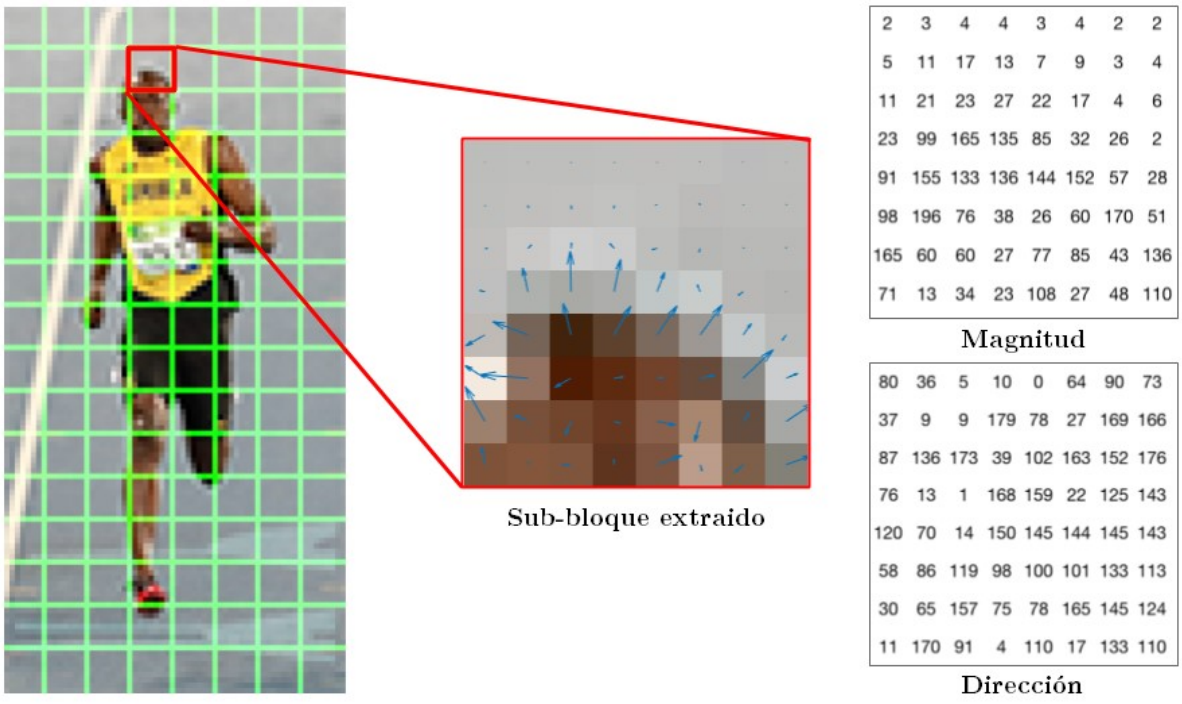

Figura A.2: Resultados del cálculo del HOG [17]

En la **figura A.2** se pueden observar los valores de dirección y magnitud del gradiente de cada píxel. Se puede observar que todos los ángulos se encuentran **entre 0 y 180 grados** en vez de entre 0 y 360 grados. Esto es llamado **gradientes sin signo**, debido a que un gradiente y su opuesto 180 grados son considerados iguales. Además, se ha demostrado empíricamente que los gradientes sin signo trabajan mejor para gran cantidad de casos que los gradientes con signo.

El histograma se generará esencialmente mediante un **vector de 9 elementos**, en los que cada elemento representa un ángulo diferente (0, 20, 40, 60… 160). El elemento del histograma es elegido **en base a la dirección** de cada píxel, y el **valor aportado depende de la magnitud** de este. Aquellos píxeles cuya dirección se encuentre entre dos elementos del histograma realizaran **una aportación proporcional** a su magnitud según el valor de su dirección. De esta manera, un elemento de dirección **80º** y magnitud **2** aportará un valor de **2** al elemento del histograma que corresponda con **80º**, mientras que un elemento de dirección **15º** y magnitud **4** aportará un valor de **1** al elemento correspondiente con **0º** y un valor **3** al correspondiente con **20º**. De esta forma se obtiene para cada sub-bloque el histograma correspondiente.

Debido a la gran sensibilidad de los gradientes a la iluminación general, antes de generar el descriptor, se procederá a una **normalización** de los histogramas calculados. De esta manera, se consigue la mayor independencia posible de las variaciones de luz. Para realizar esta normalización, se genera una ventana de **N×N sub-bloques**, y se desplaza de uno en uno hasta normalizar por completo la imagen. Esto implica una mejora debido a la consideración global de los valores de iluminación. El tamaño de **ventana más utilizado suele ser 2×2** sub-bloques, concatenando 4 histogramas de forma consecutiva para su normalización.

Finalmente, los vectores concatenados de tamaño 36x1 correspondientes a cada conjunto de sub-bloques **son concatenados generando el descriptor HOG** basado en gradientes. En la **figura A.3** se encuentra la representación del resultado obtenido del cálculo de su descriptor HOG para una imagen.

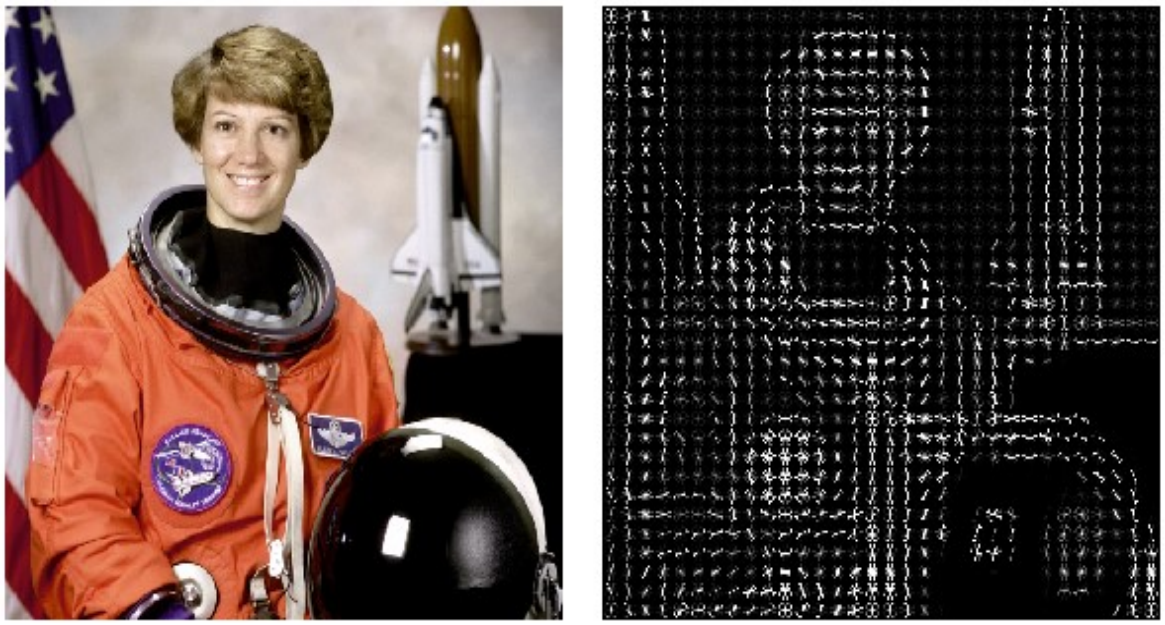

Figura A.3: Resultados del cálculo del HOG [44]

Una vez obtenido el descriptor HOG basado en histogramas de cada *frame*, es posible utilizar el **mismo método** usado en el caso de los histogramas de color. A partir de la misma **función de discriminación (A.3)**, es posible la generación de una medida de discontinuidad:

$$\boldsymbol{HogDif[i]} = \sum_{n=1}^{N} \sum_{m=1}^{M} \frac{(H_i - H_{i-1})^2}{H_i + H_{i-1}} \tag{A.8}$$

siendo **N** el número de bloques y **M** el número de elementos contenidos en cada bloque. De esta manera, se obtendrá un resultado similar al caso de la comparación anterior, **observando picos definidos** cuando la diferencia entre *frames* adyacentes sea demasiado abrupta, detectando de esta forma las transiciones abruptas o cortes en el video.

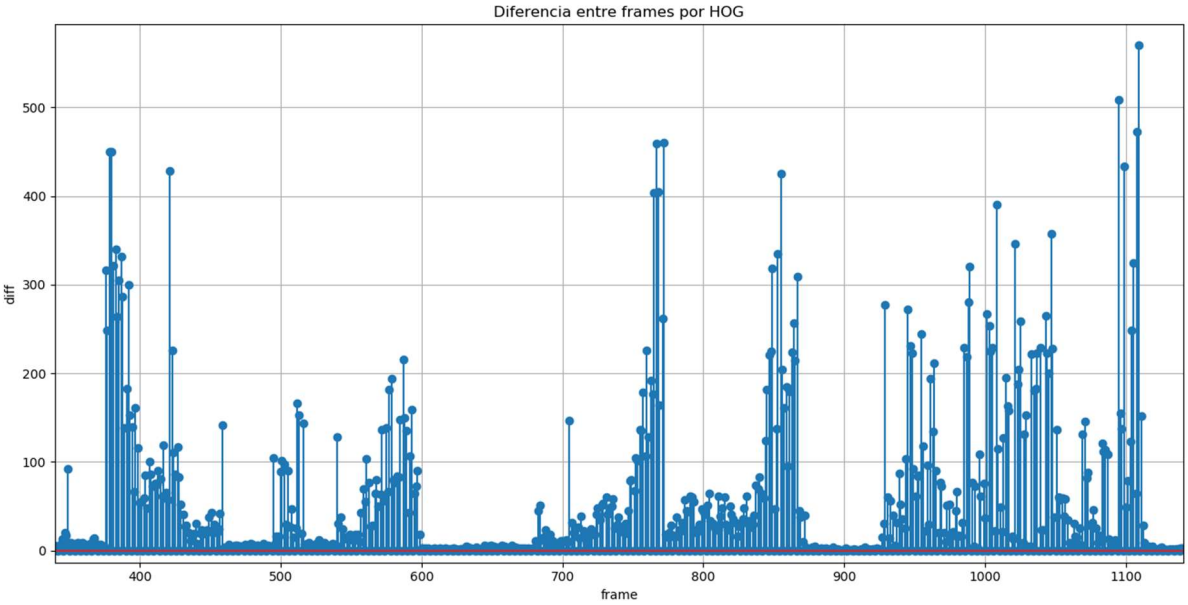

Figura A.4: Resultados de la diferencia entre histogramas de color

Anexo B

Redes Neuronales

En este anexo se mostrará con mayor detalle los fundamentos teóricos en los que se basan las redes neuronales, así como las utilizadas en este proyecto.

B.1 Introducción

Las redes neuronales están compuestas por un conjunto de unidades neuronales simples, de forma aproximadamente análoga al comportamiento observado en los axones de las neuronas de los **cerebros biológicos**. Cada unidad neuronal está conectada con muchas otras y los enlaces adyacentes entre ellas pueden incrementar o inhibir el estado de activación de las neuronas adyacentes. El objetivo de la red neuronal es resolver los problemas de la misma manera que el cerebro humano, aunque las redes neuronales son más **abstractas**. Para poder entender una red neuronal, es necesario definir la unidad que la compone, la **neurona artificial**. Una neurona es una unidad computacional con **n** entradas y una salida, tal y como se muestra en la **figura B.1**.

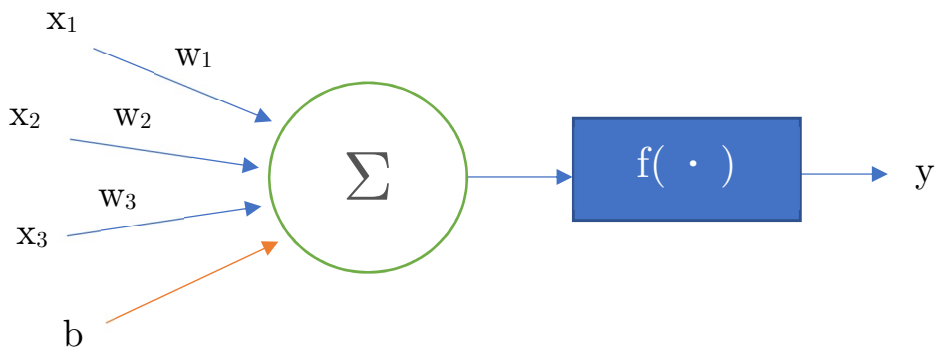

Figura B.1: Esquema general de una neurona

La neurona queda descrita por los siguientes elementos:

- **Entradas:** representadas mediante $\mathbf{x_i}$**,** son las entradas de datos externos, o las salidas de la capa anterior de neuronas.
- ***Bias*:** representado mediante **b**, es el término **constante** o de intersección de la combinación lineal, que se aplicará en el análisis teórico de las redes neuronales.
- **Pesos:** descritos por $\mathbf{w_i}$, son los **coeficientes** por los que se multiplican las entradas de la neurona.

- **Sumatorio:** Se encarga de realizar la suma del **bias** y todas las entradas multiplicadas por sus respectivos pesos y de mandar el resultado a la función de activación.
- **Función de activación:** función no lineal indicada por **f(·)**. Hay diversos tipos de función de activación. Dos de ellas son la **sigmoide** y la **tangente hiperbólica**, cuyas expresiones quedan definidas en las **ecuaciones B.1** y **B.2**, respectivamente.

$$\boldsymbol{f(z) = \sigma(z) = \frac{1}{1 + e^{-z}}} \quad \textbf{(B.1)}$$

$$\boldsymbol{f(z) = tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}} \quad \textbf{(B.2)}$$

Para el cálculo de la salida de la función de activación, se definen dos pasos para cada una. En primer lugar, se realizan **M combinaciones lineales** en función del número de las entradas. En el caso de la capa de entrada, será el número de entradas para cada neurona, mientras que, para el resto de las capas, serán las salidas de las neuronas de la capa anterior.

$$\boldsymbol{a_k = \sum_{i=1}^{D} w_{ik} x_{ik} + b_k} \quad \textbf{(B.3)}$$

siendo **D** el número de entradas de la neurona, $\mathbf{w_{ik}}$ cada uno de los pesos por los que multiplicar las entradas, $\mathbf{x_{ik}}$, y $\mathbf{b_k}$ el ***bias***, siendo **i** el identificar de la entrada y **k** el identificador de la neurona analizada. La función del ***bias***, es muy importante puesto que, al ser la parte constante de la combinación lineal, **permite desplazar la curva** de la red neuronal arriba o abajo. Esto puede ser **crítico para un aprendizaje exitoso**. Por último, la salida de esta combinación lineal $\mathbf{a_k}$, es introducida en la función de activación, tal y como puede observarse en la **ecuación B.4.**

$$\boldsymbol{y_k = f(a_k)} \quad \textbf{(B.4)}$$

B.2 Redes *Feed-forward*

Las redes *feed-forward*, también conocidas como **perceptrones** o **MLP (*Multi-Layer Perceptron*)** en caso de tener varias capas, son un tipo de redes en las que **no se produce ningún bucle** o ciclo. Este tipo de redes se encuentra entre las más **sencillas** y sirven como introducción general al resto de las redes neuronales.

Las redes neuronales están constituidas por la unión de neuronas, las cuales se encuentran **estructuradas en capas**, de tal manera que la entrada a una neurona en la capa **i** es la salida de la capa $\mathbf{i - 1}$. Las redes neuronales suelen estar formadas por una capa de entrada, una de salida y una o más **ocultas**, que son las que definen la complejidad de esta, como se puede observar en la **figura B.2**.

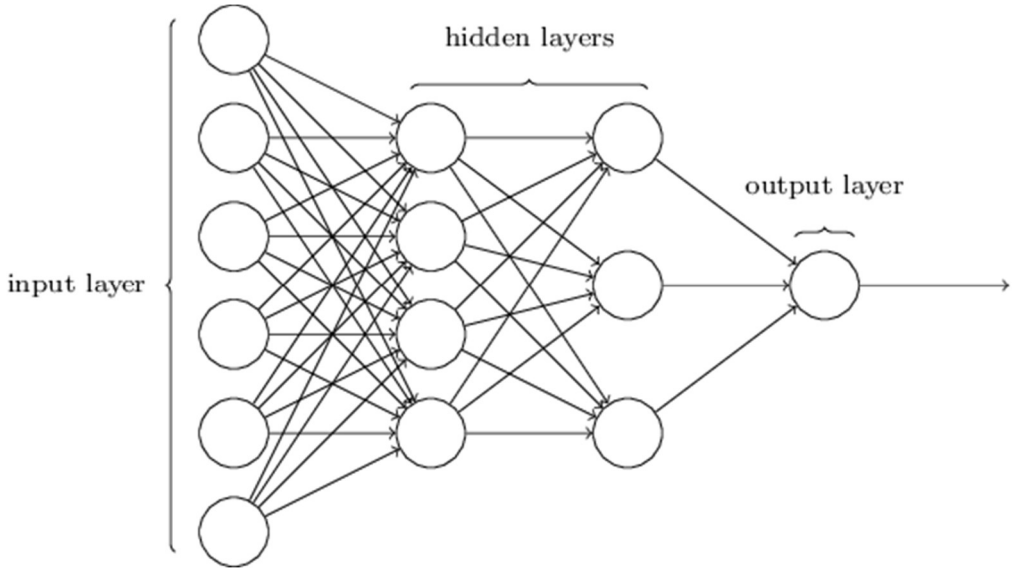

Figura B.2: Ejemplo de una red neuronal [45]

Las redes multicapa utilizan una gran variedad de técnicas de aprendizaje, entre las que destaca el ***backpropagation algorithm***. Los valores de salida son comparados con la respuesta correcta para computar el error con una función ya predefinida. De esta forma el error es realimentado por la red. A través de esta información, el algoritmo ajusta las ponderaciones de cada conexión para reducir el valor de la función de error. Después de repetir este proceso durante un número de ciclos lo suficientemente grande, la red convergerá a un estado en el que **el error producido sea lo menor posible**. Este ajuste de las ponderaciones se realiza mediante el denominado **descenso de gradiente**, en el que se busca la derivada de error en función de los parámetros entrenables **w** y **b**, actualizando iterativamente estos parámetros en la dirección de máxima pendiente.

B.3 Redes Convolucionales (CNN)

Las **redes neuronales convolucionales (CNN)** [19] son muy similares a las redes neuronales *feed-forward*. Como se describió antes, una CNN simple es una secuencia de capas, y cada capa realiza la transformación de un volumen mediante activaciones, a otro a través de una función diferenciable. Normalmente, son usadas **tres tipos principales** de capas para construir las arquitecturas de las CNN: **Convolucionales**, ***Pooling*** y ***Fully-Connected***. Al apilar estos tres tipos diferentes es posible construir la arquitectura deseada. Además de estos tres tipos de capas, también es común encontrar la capa de activación ***ReLu*** o capa **rectificadora**, definida en la **ecuación B.5.**

$$\boldsymbol{f(x) = x^{+} = max(0, x)} \tag{B.5}$$

Esta función de activación permite obtener mejores resultados en el entrenamiento de redes neuronales profundas comparado con la función **sigmoide** y es más práctica que la **tangente hiperbólica**. La función rectificadora es actualmente la función de activación **más utilizada** en el campo de las redes neuronales profundas. Esta capa **no modifica las dimensiones** del volumen de entrada.

La **capa convolucional** corresponde con el núcleo de la CNN y es la encargada de la mayor carga computacional. Se caracteriza por: su **profundidad**, siendo esta el número de filtros por la capa, el **paso**, definirá la forma en la que desplazaremos el filtro a través de la imagen y el ***zero padding***, utilizado para rellenar los marcos de la entrada con ceros y poder controlar el tamaño de los volúmenes de salida. Debido a la gran cantidad de conexiones necesarias, esta se conecta únicamente a una región del volumen de entrada, llamada **campo receptivo** de la neurona. Las conexiones son **locales en espacio** (alto y ancho) pero **completas en profundidad**. También es posible reducir drásticamente el número de parámetros. Si una característica es útil para realizar el cálculo en cierta posición espacial, también podría serlo para otra diferente, denotándose esta superficie de dos dimensiones como **corte de profundidad**. Este comparte los mismos pesos y *bias*. Es por esto por lo que comúnmente se denota al **conjunto de los pesos** como **filtro**. En ciertas ocasiones, es común relajar estas restricciones para otorgar mayor flexibilidad a las capas, llamándose **capas localmente conectadas**.

Es muy común en las arquitecturas insertar entre sucesivas capas convolucionales capas de ***Pooling***, cuya función es reducir de forma progresiva el tamaño para **disminuir la cantidad de parámetros** y **controlar el sobreentrenamiento**. Estas capas operan de forma independiente en cada **corte de profundidad** de la red y los redimensiona espacialmente utilizando para ello la función ***max***, siendo la más común la aplicación de **filtros 2×2**, descartando el 75% de las activaciones.

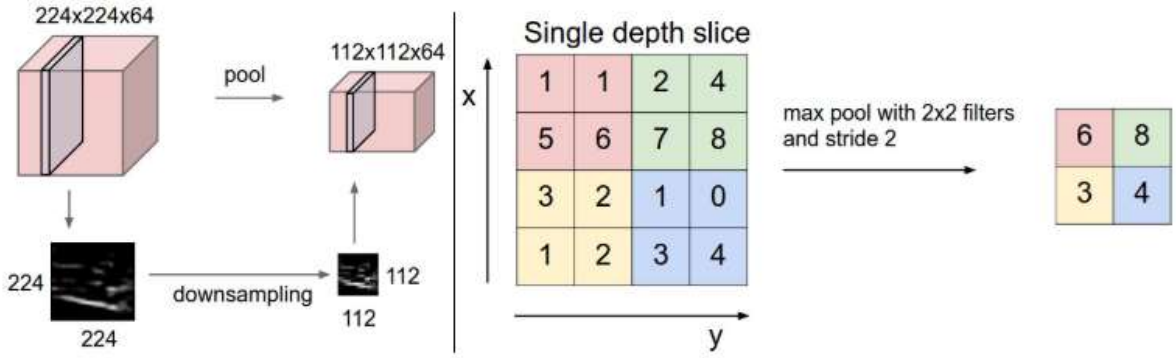

Figura B.3: Ejemplo de muestreado en la capa de *pooling* [19]

También es posible encontrar en estas arquitecturas **capas de normalización**, con la intención de imitar esquemas de inhibición observados en las redes neuronales biológicas. Sin embargo, estas capas son cada vez menos utilizadas debido a que en la práctica ha sido comprobado que sus contribuciones son mínimas. La arquitectura finalizará con **una o varias capas *fully-connected*** de igual diseño que las redes ***feed-forward*** regulares. Poseen todas sus neuronas interconectadas y traducen las características obtenidas del volumen de entrada en las puntuaciones finales de clase. Las CNN transforman la imagen original, pasando de los píxeles iniciales a la puntuación final de clases.

Es destacable el hecho de que no todas las capas de la arquitectura contengan parámetros. Los parámetros de las **capas de convolución** y **fully-*connected*** serán entrenados mediante el uso del **descenso de gradiente**. De forma que, las puntuaciones obtenidas por el cálculo de la CNN sean lo más consistentes con las etiquetas asociadas a cada imagen del conjunto de entrenamiento.

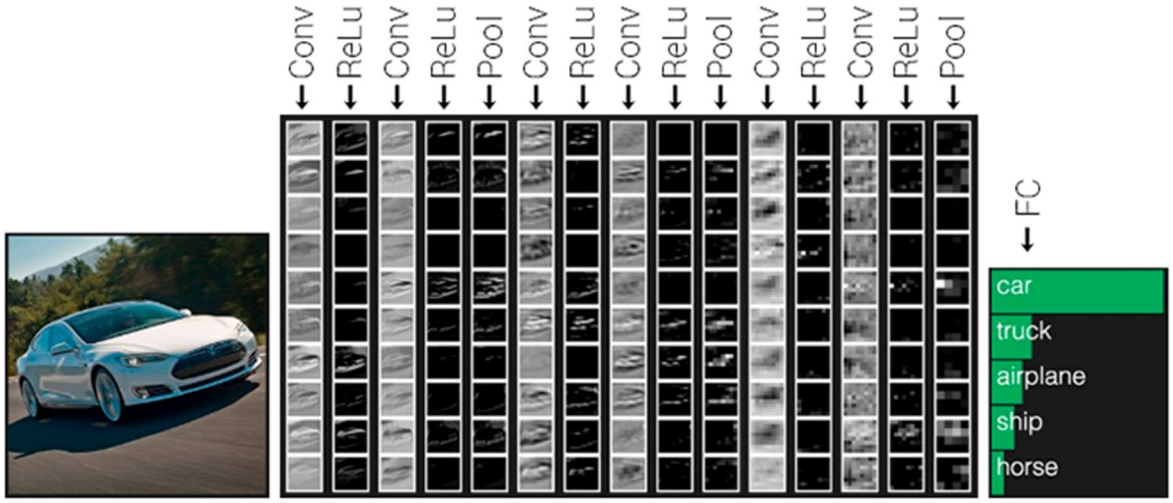

Figura B.4: Ejemplo de arquitectura completa de una CNN [19]

B.3.1 Redes ResNet

Las **redes neuronales residuales** o **ResNets** [21], son un tipo de redes convolucionales. Como el propio nombre de la red indica, este tipo introduce el **aprendizaje residual**. Con el tiempo, las redes convolucionales han ido progresando y mejorando en el campo de la clasificación de imágenes. Estas han ido ganando cada vez **mayor profundidad**, apilando un mayor número de capas que les permita la resolución de procesos más complejos y el aumento de la precisión en las tareas de reconocimiento y clasificación.

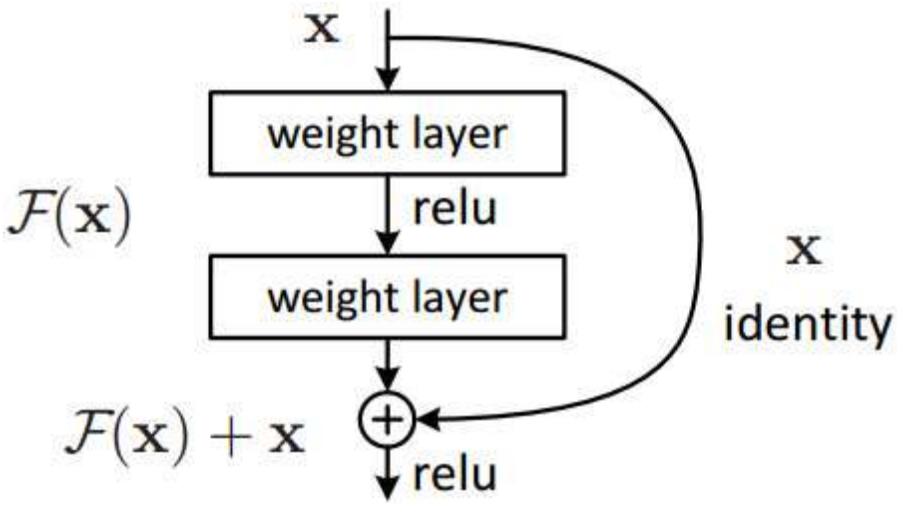

Figura B.5: Aprendizaje residual de una ResNet [47]

Sin embargo, cuanto mayor es esta profundidad, **mayor es la complejidad de su entrenamiento**, provocando a su vez una saturación y **degradación de la precisión** conseguida. El aprendizaje residual intenta solucionar ambos problemas. En general, en una CNN se apilan varias capas y se entrenan para la tarea en cuestión. La red aprende varias

características de nivel bajo/medio/alto al final de sus capas. En el **aprendizaje residual**, en vez de tratar de aprender algunas características, se trata de aprender algo del residuo.

El **residuo** [20] se puede entender simplemente como la resta de la característica aprendida de la entrada de esa capa. ResNet hace esto conectando directamente la entrada de la capa con alguna de las capas inferiores. De esta forma, el entrenamiento de estas redes resulta mucho más sencillo que el de las redes convolucionales convencionales. Además, consigue resolver el problema de la degradación de la precisión, lo que **permite profundizar más** de que lo que una red convencional permitiría.

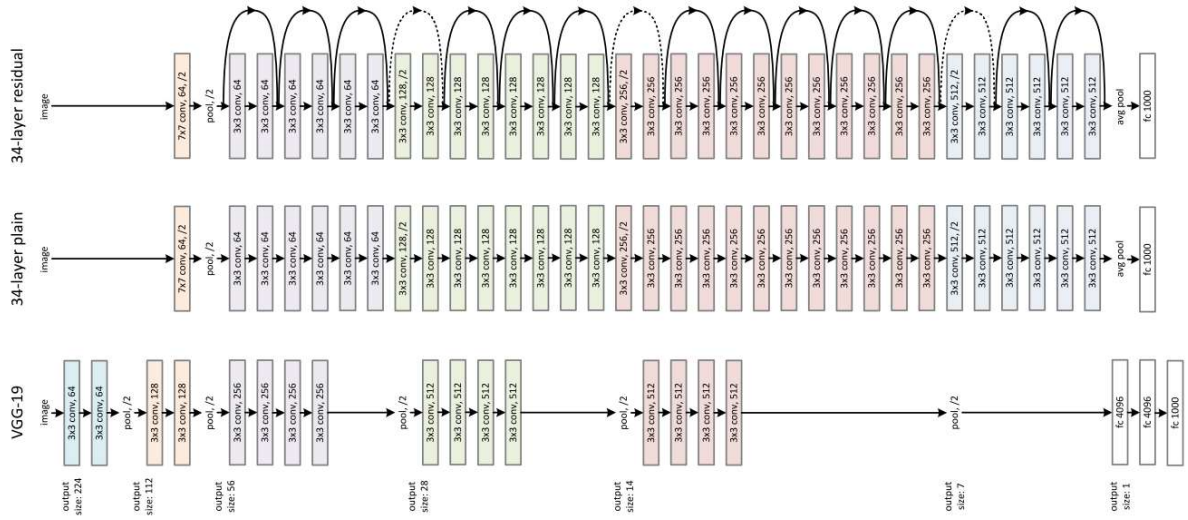

Figura B.6: Comparación con arquitectura ResNet [46]

B.4 Redes Neuronales Recurrentes (RNN)

Las redes recurrentes pueden ser pensadas como múltiples copias de la misma red, sin realimentación, cada una pasándole un mensaje a su sucesiva, tal y como se muestra en la **figura B.8**. Son entendidas **como cadenas de varias redes *feed-forward* unidas**. De hecho, su entrenamiento se basa en esta idea, denominada ***backpropagation through time***.

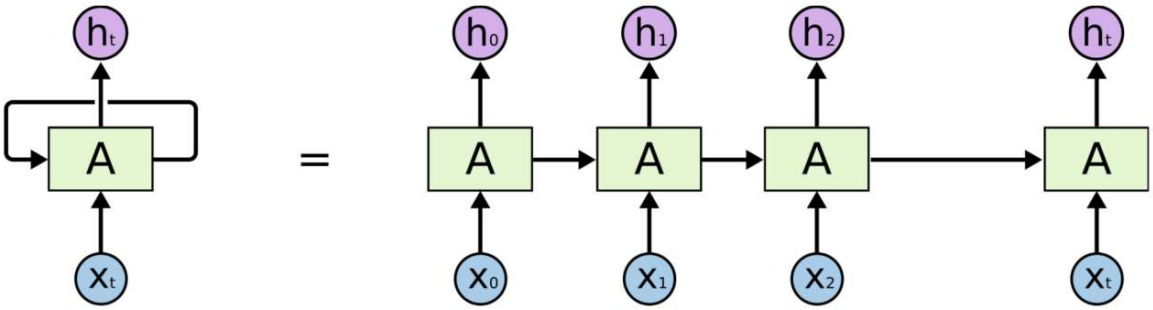

Figura B.7: Ejemplo del esquema genera de una RNN [24]

Se han obtenido grandes progresos aplicando estas redes a una gran variedad de problemas: reconocimiento del habla, modelado del lenguaje, traducción, subtitulado… Gran parte de este éxito se debe al uso de las **LSTM *(Long Short Term Memory)***, un tipo especial de red recurrente que funciona, en muchas ocasiones, mejor que la versión estándar. Aunque teóricamente las redes recurrentes deben poder gestionar también dependencias de larga duración, en la práctica no sucede así, ya que conforme el *gap* entre la información relevante y el punto donde se necesita es mayor, las RNNs funcionan peor. Además, se **gana estabilidad** respecto a los desvanecimientos y explosiones del gradiente.

B.4.1 Redes LSTM

Las redes **LSTM** [23] son un tipo de redes neuronales recurrentes, capaces de asimilar dependencias de larga duración, es decir, que lo aprendido en un instante de tiempo anterior pueda ser utilizado posteriormente. Fueron propuestas por ***Hochreiter*** y están diseñadas específicamente para evitar el problema de las largas dependencias con el tiempo. Esto provoca una mayor complejidad en las neuronas, tal y como se puede observar en la **figura B.9** en la que aparece representada una celda de memoria LSTM.

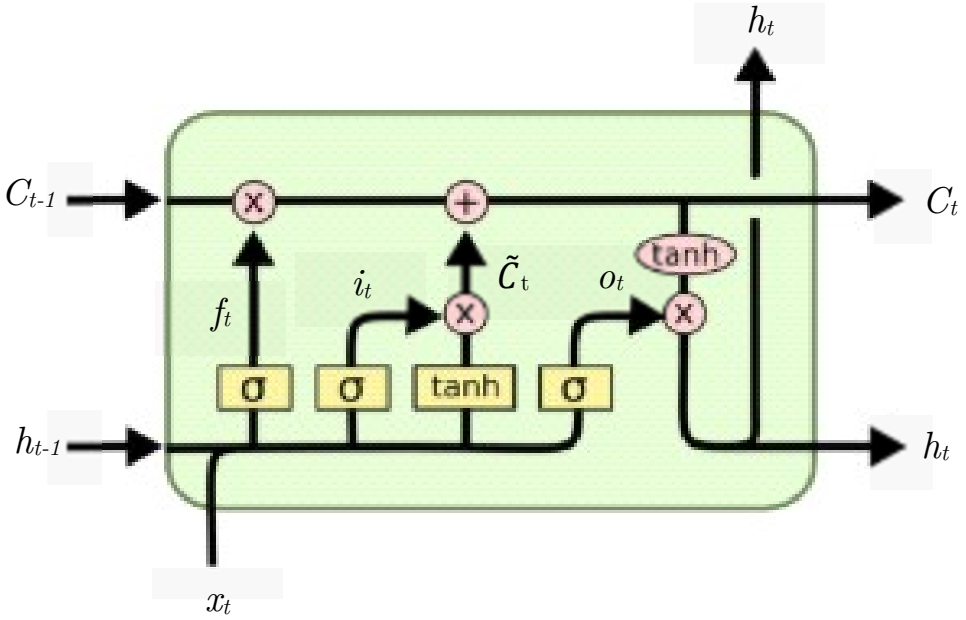

Figura B.8: Celda de memoria de una LSTM [24]

Mientras que una RNN estándar consiste en una única función de activación, que se utiliza para obtener la salida a partir de la entrada, en una LSTM [24], se añaden una serie de pasos intermedios que se congregan la celda de memoria. La celda tiene la habilidad de **añadir o eliminar información de esta**, regulada cuidadosamente por unas estructuras denominadas **puertas**. Las puertas se encuentran compuestas por una **capa sigmoide y un operador multiplicación**. La sigmoide muestra salidas comprendidas entre cero y uno, describiendo la proporción guardada de la salida. Una neurona LSTM posee **tres puertas** para controlar y proteger el estado de la celda:

- ***Forget gate:*** Decide la cantidad de información del estado anterior que es **eliminada del estado** de la celda. Su funcionamiento está definido por la **ecuación B.6**.

$$\boldsymbol{f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big)} \quad \textbf{(B.6)}$$

- ***Input gate:*** Decide la cantidad de información nueva que se **almacena en el estado**. Posee dos partes, en primer lugar, una capa **sigmoide** decide los valores a actualizar, en segundo lugar, una capa **tangente hiperbólica** crea un nuevo vector con los valores candidatos, $\widetilde{\boldsymbol{C}}_{\boldsymbol{t}}$ que serán añadidos al estado. En el siguiente paso se combinarán estos dos elementos **generando una actualización** del estado. Ambos pasos quedan descritos por las **ecuaciones B.7** y **B.8**.

$$\boldsymbol{i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)} \quad \textbf{(B.7)}$$

$$\boldsymbol{\widetilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)} \quad \textbf{(B.8)}$$

  Tras decidir la información que será olvidada del estado y la que será añadida, se procede a **actualizar el valor** de este con los valores previamente calculados, escalados según el valor de las sigmoides.

$$\boldsymbol{C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t} \quad \textbf{(B.9)}$$

- ***Output gate:*** Decide la cantidad de información que irá **a la salida**. Esta salida está basada en una **versión filtrada del estado** de la celda. Primero se decidirá mediante una capa sigmoide las partes del estado que corresponderán a la salida. Finalmente, el estado se introduce en una capa **tangente hiperbólica** que será multiplicado por el **resultado de la capa sigmoide**.

$$\boldsymbol{o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)} \quad \textbf{(B.10)}$$

$$\boldsymbol{h_t = o_t * tanh(C_t)} \quad \textbf{(B.11)}$$

Gracias a este mecanismo, las redes **LSTM** tienen la habilidad de añadir, preservar o eliminar la información. Finalmente, cabe destacar que este tipo de redes son muy útiles a la hora de manejar **grandes cantidades de datos**, puesto que hay mayores distancias temporales y son más potentes que las redes *feed-forward*, aunque esto suponga la utilización de arquitecturas mucho más complejas.

B.4.2 Redes BiLSTM

Las redes **BiLSTM** [40] son una **evolución de las LSTM**, cuya principal ventaja es que son **bidireccionales**. Son formadas al combinar dos redes LSTM recorriendo **una dimensión en sentidos opuestos**, generalmente la temporal. Por tanto, pueden aprender información tanto de instantes pasados como futuros. Esto conlleva una mayor complejidad, pero también aporta unas mejores prestaciones a costa de la **imposibilidad de operar en tiempo real**.

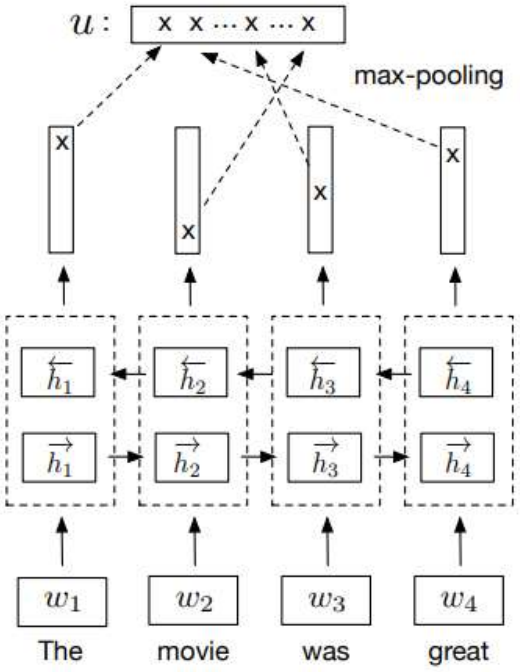

Figura B.9: Red BiLSTM con *max-pooling* [40]

Anexo C

Procesado del Lenguaje Natural

En este anexo se mostrará con mayor detalle los fundamentos teóricos en los que se basan las técnicas de procesado de lenguaje natural que han sido utilizadas en este proyecto.

C.1 *Word Embeddings*

Los *word embeddings* tienen la propiedad de que aquellos con significados relacionados se encuentran a **menor distancia** entre ellos que los pertenecientes a palabras con significados completamente diferentes. Esto se debe a que la distribución de la representación está basada en el uso de las relaciones entre las palabras, lo que **permite capturar y representar su significado.**

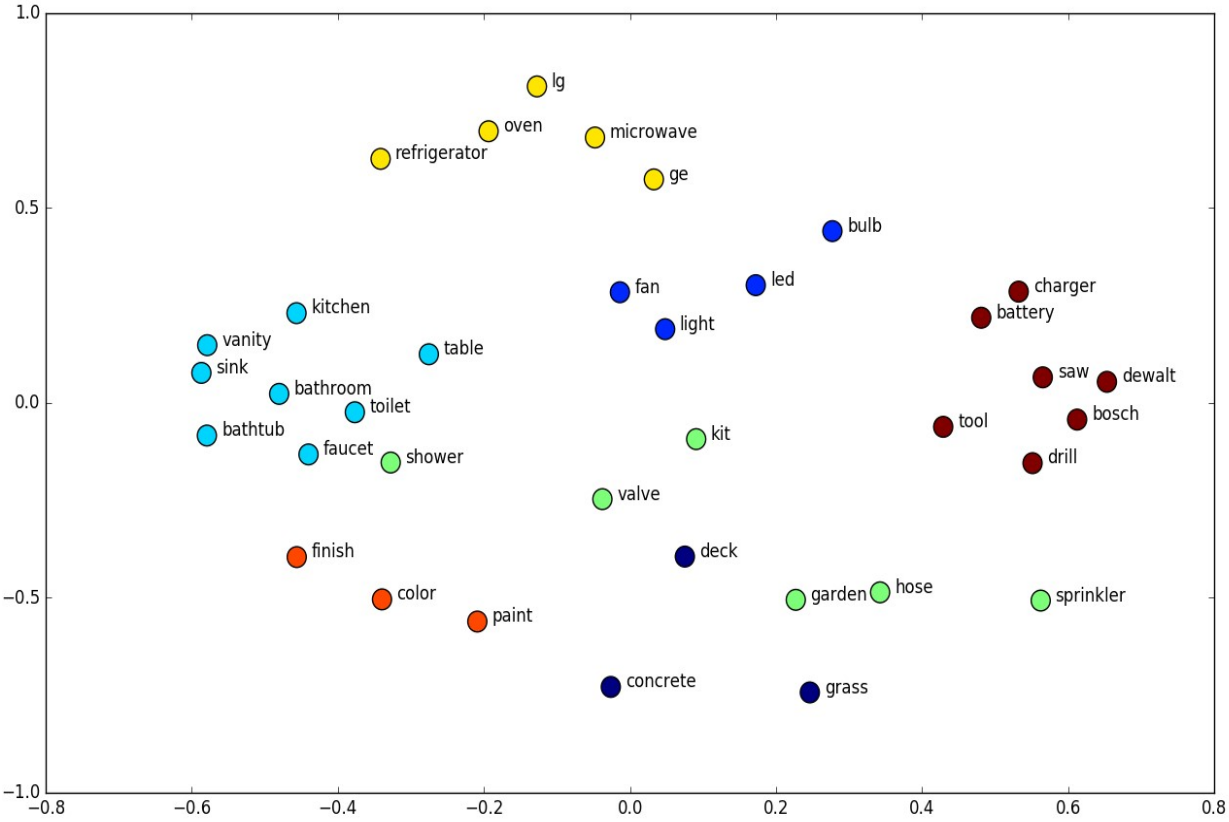

Figura C.1: Representación de palabras mediante sus *embeddings* [48]

Los *word embeddings* también presentan otras propiedades interesantes. Las analogías entre palabras pueden ser **codificadas mediante la diferencia** entre los vectores de dos palabras, pudiendo obtener, por ejemplo, un vector diferenciador constante hombre-mujer como se indica en las **ecuaciones C.1** y **C.2**.

$$\boldsymbol{W}(\text{"}\boldsymbol{woman}\text{"}) - \boldsymbol{W}(\text{"}\boldsymbol{man}\text{"}) \simeq \boldsymbol{W}(\text{“}\boldsymbol{aunt}\text{"}) - \boldsymbol{W}(\text{“}\boldsymbol{uncle}\text{"}) \quad \textbf{(C.1)}$$

$$\boldsymbol{W}(\text{"}\boldsymbol{woman}\text{"}) - \boldsymbol{W}(\text{"}\boldsymbol{man}\text{"}) \simeq \boldsymbol{W}(\text{“}\boldsymbol{queen}\text{"}) - \boldsymbol{W}(\text{“}\boldsymbol{king}\text{"}) \quad \textbf{(C.2)}$$

Existen diferentes métodos para extraerlos, pudiendo diferenciar principalmente entre dos:

- ***Embeddings* por frecuencia:** Se basa en métodos deterministas, realizando la vectorización mediante la **contabilización de repeticiones** de cada una de las palabras según un diccionario previamente generado y su posterior procesado. Entre estos podemos encontrar la generación por **vectores de conteo**, la vectorización ***tf-idf* (*Term frequency – Inverse document frequency)***, que trata de penalizar aquellas palabras que se repitan un gran número de veces, y el uso de **matrices de coocurrencia**, basándose en la idea de que palabras similares tienden a ocurrir en contextos similares. Muchas de estas técnicas también se utilizan en el **cálculo de LSA** como veremos más adelante.

- ***Embeddings* por vector basado en predicción:** Están basados en el uso de redes neuronales, y a diferencia de los métodos deterministas, proveen de **probabilidades** a las palabras y permiten el uso de analogías y similitudes entre estas. Se basan en la aplicación del algoritmo ***word2vec*** [**54**], que es combinación de dos métodos: **CBOW *(Continous bag of words)*** y el modelo de ***Skip-gram***. Ambas técnicas aprenden y modifican sus pesos para poder realizar una representación numérica de las palabras.

C.2 Análisis Semántico Latente (*LSA*)

La aplicación del algoritmo LSA para la generación de resúmenes se puede reducir en 3 pasos principales: la **generación de la matriz** de entrada a partir un vocabulario, la **descomposición** SVD y la **selección** de descripciones.

Primero se construirá una matriz que contenga la contabilización de cada palabra en cada documento, de igual manera que en la **generación de *embeddings* por frecuencia**. La forma de rellenar los valores de la matriz de entrada **influye** en el resultado de la descomposición SVD, pudiendo diferenciar los siguientes métodos:

- **Frecuencia de las palabras**: la celda es rellenada en base a la **frecuencia** de repetición de la palabra en la descripción.
- **Representación binaria**: la celda es rellenada con 0/1 dependiendo de la **presencia** de la palabra en la descripción.
- ***Tf-idf***: la celda se rellena con el valor *tf-idf* [42] de la palabra. Valores altos significan que la palabra es **muy frecuente en la frase, y menos frecuente en la colección**

de descripciones, indicando que esa palabra es mucho más representativa para esa frase que otras.

- **Entropia**: la celda es rellenada mediante el valor de la entropía de la palabra, indicando **cuanta información** aporta a la descripción.
- **Raiz**: la celda es rellenado con 0/1 si la **raíz de la palabra es un nombre**.
- ***Tf-idf* modificado**: versión **modificada de *tf-idf*** para suprimir el ruido, elimina los valores de aquellas celdas con puntuaciones **menores a la media**.

Tras esto, utilizando el procesado **SVD** (**Singular Value Decomposition)** sobre la matriz se reducirán el número de filas mientras se **preserva la estructura de similitud** entra las columnas. Este es un **algoritmo cuya complejidad aumenta** conforme se incrementa el tamaño y la no homogeneidad de las matrices, **degradando el rendimiento** de este. Por ello, se ha reducido previamente el tamaño del vocabulario mediante el uso de aproximaciones, como la no inclusión de palabras vacías (artículos, pronombres, preposiciones…) o el uso de las raíces de las palabras.

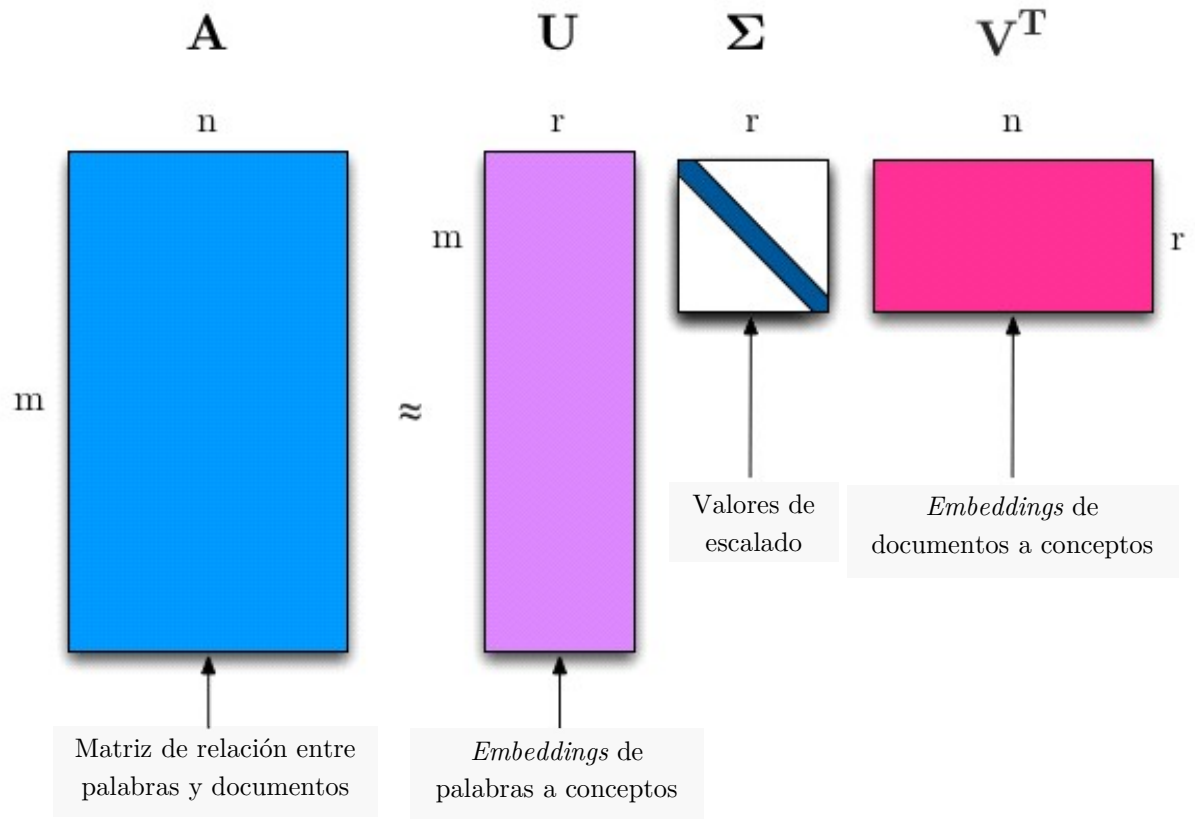

Figura C.2: Descomposición SVD de la matriz A [49]

El procesado de la matriz se realiza mediante la descomposición en valores singulares SVD. Permite la **factorización de una matriz** como producto de otras tres:

$$\boldsymbol{A_{m\times n} = U_{m\times r}\Sigma_{r\times r}(V_{n\times r})^T} \qquad \textbf{(C.3)}$$

donde **A** es la matriz de entrada, **U** es una matriz cuyas columnas son los vectores singulares a izquierdas de **A, Σ** una matriz diagonal positiva que contiene los valores singulares ordenados de la matriz **A**, y la matriz **V** cuyas filas contienen los vectores singulares a derechas de **A**. Cada una de las matrices obtenidas tendrá una función diferente, siendo **U** la que relacione las palabras con los conceptos, **Σ** captura la importancia de cada uno de estos y **V** la que relaciona los documentos con los conceptos.

A partir del resultado de la composición SVD, se procederá a la **selección de las descripciones de mayor relevancia** mediante el uso de diferentes tipos de algoritmos:

- ***Gong and Liu***: $\mathbf{V^T}$ es utilizada para la extracción de las descripciones, en la que el orden de filas indica la importancia de los conceptos extraídos, siendo la primera fila la más importante. Se selecciona la **descripción que mayor relación tenga con cada concepto en orden descendente**, hasta alcanzar las $\mathbf{N_r}$ del resumen.
- ***Steingber and Jezek***: usa tanto **Σ** como **V**, utilizando las filas de **V** como **vector de longitud** de la frase. Estos vectores de longitud son multiplicados por **Σ**, enfatizando más en los conceptos más importantes. Por último, se calcula el sumatorio de cada vector, y se eligen las **descripciones con un mayor valor de longitud**. Trata de cubrir algunas de las deficiencias del algoritmo ***Gong and Liu***.
- ***Murray***: usa tanto **Σ** como $\mathbf{V^T}$, pudiendo seleccionar más de una descripción por concepto. Se toman aquellas descripciones con un mayor valor según $\mathbf{V^T}$, pudiendo elegir **tantas descripciones por cada concepto como el porcentaje** del valor singular indique.
- ***Cross method*: Es una extensión del algoritmo *Steingber and Jezek*,** aunque tan **solo** utiliza $\mathbf{V^T}$ para realizar la selección. Toma **un paso adicional de pre-procesado** tras la generación de las matrices, calculando la media de cada una de las filas de $\mathbf{V^T}$ y discriminando las celdas menores que la media. Se eliminan las descripciones menos importantes mientras que se mantienen las más relacionadas con cada concepto. Tras esto, se procede de igual manera que el algoritmo ***Steingber and Jezek*** con una modificación. Se realizará el sumatorio de cada columna de $\mathbf{V^T}$, eligiendo las **descripciones con un mayor valor de longitud**.
- ***Topic method***: Trata de encontrar los principales conceptos y subconceptos. Para ello ejecuta el **mismo pre-procesado** que el método anterior. Después genera una **matriz concepto×concepto** basándose en $\mathbf{V^T}$ para el cálculo de la importancia de cada uno de estos, eligiendo el principal de esta manera. Finalmente, las descripciones serán elegidas siguiendo los criterios del **algoritmo *Gong and Liu.***

Anexo D

Métricas

Este anexo muestra las principales métricas utilizadas: **BLEU** (*Bilingual evaluation understudy*), **ROUGE** (*Recall-Oriented Understudy for Gisting Evaluation*), **METEOR** (*Metric for Evaluation of Translation with Explicit Ordering)* **y CIDEr** (*Consensus-based Image Description Evaluation*):

D.1 BLEU

BLEU es un popular traductor de métricas que analiza las coocurrencias de n-gramas entre el candidato y las referencias. Para ello, calcula la **precisión acotada de los n-gramas a nivel del corpus** entre frases. La métrica limita el número de veces que un n-grama puede ser contado al máximo número de veces que es observado en una única referencia. BLEU ha mostrado un **buen rendimiento** para comparaciones a **nivel del corpus** cuando un alto número de coincidencias entre n-gramas existe, sin embargo, a nivel de frase, las coincidencias entre n-gramas para valores altos de **n** rara vez ocurren. Por ello BLEU obtiene **rendimientos pobres** cuando se comparan **frases individuales**.

D.2 ROUGE

ROUGE es un **conjunto de métricas** diseñadas para evaluar algoritmos para resúmenes de texto. Se puede diferenciar en tres variantes diferentes: ROUGE_N realiza un **cálculo simple sobre los n-gramas recordado**s sobre los resúmenes de referencia dada una frase candidata, ROUGE_L utiliza una medida basada en la **subsecuencia común más larga** (LCS), un conjunto de palabras compartido entre dos frases diferentes el cual ocurre en el mismo orden, y ROUGE_S utiliza **bigramas *skip*** en vez de n-gramas o LCS, siendo un bigrama *skip* un par de palabras ordenadas en una frase.

D.3 METEOR

METEOR es calculado realizando un **alineamiento entre las palabras del candidato y las referencias**, con un objetivo de correspondencia 1:1. Este alineamiento es calculado mientras se minimiza el número de bloques y de palabras contiguas e idénticamente ordenadas. METEOR realiza una reducción de las palabras a sus raíces.

D.4 CIDEr

CIDEr mide el consenso entre los *captions* de una imagen mediante el **cálculo del peso *tf-idf*** para cada n-grama. *Tf-idf* tiene en cuenta el término *tf,* que se incrementa para aquellos **n-gramas que ocurren frecuentemente** en las referencias, y el término *idf*, que produce una **medida de la importancia de una palabra**, penalizando aquellas más populares que son propensas a aportar una menor cantidad de información. CIDER posee una variación, CIDEr-D, el cual es **más robusto** al *gaming* (fenómeno en el que una oración pobremente juzgada por personas tiende a obtener puntuaciones altas con una métrica automatizada). Para ello agrega una penalización gaussiana basada en la duración de la métrica. Al igual que BLEU y ROUGE, CIDEr-D no reduce las palabras a sus raíces.

Bibliografía

[1] Self-critical sequence Training for Image Captioning. *https://github.com/ruotianluo/self-critical.pytorch*, 2018.

[2] Image Captioning with Spatial Attention in Keras. *https://github.com/amaiasalvador/imcap_keras*, 2017.

[3] NeuralTalk2. *https://github.com/karpathy/neuraltalk2*, 2016.

[4] InferSent. *https://github.com/facebookresearch/InferSent*, 2018.

[5] COCO API. *https://github.com/cocodataset/cocoapi*, 2018.

[6] Keras: The Python Deep Learning library, sitio web: Página principal. *https://keras.io/*, 2018.

[7] Python, sitio web: Página principal. *https://www.python.org/*, 2018.

[8] Pytorch, sitio web: Página principal. *https://pytorch.org/*, 2018.

[9] Brandwatch, 36 estadísticas fascinantes de YouTube para 2016, *https://www.brandwatch.com/es/blog/36-estadisticas-youtube-2016/* ,2016.

[10] COCO, Common Objects in Context. *http://cocodataset.org/#home*, 2018.

[11] Antoine Manzanera. Indexación de imágenes y vídeos. *http://perso.ensta-paristech.fr/~manzaner/Download/Tutorials/Curso_Indexacion_Bogota08.pdf*, 2008.

[12] Yanio Hernández Heredia, José Ortiz Rojas, Ruber Hernández García and José María González Linares. Descriptores temporales-espaciales en la detección automática de información audiovisual. *http://www.redalyc.org/html/1814/181423798003/*, 2012

[13] Javier Jorge Cano. Clasificación de vídeos mediante Redes Neuronales Artificiales. *https://riunet.upv.es/bitstream/handle/10251/64848/TrabajoFinalMasteJavierJorgeCano.pdf?sequence=1*, 2015

[14] Jordi Delcor Ballesteros and Verónica Pérez Noriega. Descripción, indexación, búsqueda y adquisición de secuencias de vídeo mediante descriptores MPEG-7. *https://upcommons.upc.edu/bitstream/handle/2099.1/3855/54955-1.pdf*, 2006.

[15] Mayu Otani, Yuta Nakashima, Esa Rahtu, Janne Heikkilä, and Naokazu Yokoya. Video Summarization using Deep Semantic Features. *https://arxiv.org/pdf/1609.08758.pdf*, 2016.

[16] Gautam Pal, Dwijen Rudrapaul, Suvojit Acharjee, Ruben Ray, Sayan Chakraborty, and Nilanjan Dey. Video Shot Boundary Detection: A Review. *https://www.researchrese.net/publication/226911479_Video_Shot_Boundary_Detection_A_Review,* 2015.

[17] Satya Mallick. Learn OpenCV: Histogram of Oriented Gradient. *https://www.learnopencv.com/histogram-of-oriented-gradients/*, 2016.

[18] Diego Álvarez Gutiérrez and Ignacio Viñals Bailo. Detector de actividad vocal para Diarización mediante redes neuronales en entornos Broadcast. *https://deposita.unizar.es/record/32674?ln=es*, 2017.

[19] Convolutional Neural Networks (CNNs / ConvNets). *http://cs231n.github.io/convolutional-networks/.*

[20] Kaiming He. Deep Residual Networks. Deep Learning Gets Way Deeper. *https://icml.cc/2016/tutorials/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf*, 2016.

[21] Kartik Podugu. Quora, What is the deep neural network known as "ResNet-50"? *https://www.quora.com/What-is-the-deep-neural-network-known-as-%E2%80%9CResNet-50%E2%80%9D*, 2018.

[22] Chris Olah and Shan Carter. Attention and Augmented Recurrent Neural Networks. *https://distill.pub/2016/augmented-rnns/*, 2016.

[23] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". Neural Computation. 9 (8): 1735–1780, 1997.

[24] Christopher Olah. Understanding LSTM Networks. *http://colah.github.io/posts/2015-08-Understanding-LSTMs/*, 2015.

[25] Anantharaman Palacode Narayana Iyer. Word representation: SVD, LSA, Word2Vec. *https://es.slideshare.net/ananth/word-representation-svd-lsa-word2vec*, 2016.

[26] An Intuitive Understanding of Word Embeddings: From Count Vectors to Word2Vec. *https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/*, 2017.

[27] Mo Yu and Mark Dredze. Learning Composition Models for Phrase Embeddings. *http://aclweb.org/anthology/Q15-1017*, 2015.

[28] Susan T. Dumais. "Latent Semantic Analysis". Annual Review of Information Science and Technology. 38: 188–230, 2015.

[29] Hong Shao, Yang Qu and Wencheng Cu. Shot Boundary Detection Algorithm Based on HSV Histogram and HOG Feature. *https://www.atlantis-press.com/php/download_paper.php?id=25839347*, 2015.

[30] Jordi Mas and Gabriel Fernandez. Video shot boundary detection based on color histogram. *https://www-nlpir.nist.gov/projects/tvpubs/tvpapers03/ramonlull.paper.pdf.*

[31] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick and Piotr Dollar. Microsoft COCO: Common Objects in Context. *https://arxiv.org/pdf/1405.0312.pdf*, 2015.

[32] Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat and Barbara Plank. Automatic Description Generation from Images: A Survey of Models, Datasets, and Evaluation Measures. *https://arxiv.org/abs/1601.03896*, 2017.

[33] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross and Vaibhava Goel. Self-critical Sequence Training for Image Captioning. *https://arxiv.org/pdf/1612.00563.pdf*, 2017.

[34] Jiasen Lu, Caiming Xiong, Devi Parikh and Richard Socher. Knowing When to Look: Adaptive Attention via A Visual Sentinel for Image Captioning. *https://arxiv.org/abs/1612.01887*, 2017.

[35] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam Saurabh Gupta, Piotr Dollar and C. Lawrence Zitnick. Microsoft COCO Captions: Data Collection and Evaluation Server. *https://arxiv.org/pdf/1504.00325.pdf*, 2015.

[36] Samuel R. Bowman, Gabor Angeli, Christopher Potts and Christopher D. Manning. A large annotated corpus for learning natural language inference. *https://nlp.stanford.edu/pubs/snli_paper.pdf*, 2015.

[37] The Stanford Natural Language Inference (SNLI) Corpus. *https://nlp.stanford.edu/projects/snli/.*

[38] Jeffrey Pennington, Richard Socher and Christopher D. Manning. GloVe: Global Vectors for Word Representation. *https://nlp.stanford.edu/pubs/glove.pdf.*

[39] sklearn.manifold.TSNE. *http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html.*

[40] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault and Antoine Bordes. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. *https://arxiv.org/abs/1705.02364*, 2018.

[41] Singhal and Amit. "Modern Information Retrieval: A Brief Overview". Bulletin of the IEEE Computer Society Technical Committee on Data Engineering 24 (4): 35–43, 2001.

[42] Cambridge University Press. Term frequency and weighting. *https://nlp.stanford.edu/IR-book/html/htmledition/term-frequency-and-weighting-1.html*, 2008.

[43] Makbule Gulcin Ozsoy, Ferda Nur Alpaslan and Ilyas Cicekl. Text summarization using Latent Semantic Analysis. *https://www.researchgate.net/publication/220195824_TeTe_summarization_using_Latent_Semantic_Analysis*, 2011.

[44] Histogram of Oriented Gradients. *http://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog.html.*

[45] Feed Forward Networks. *https://transcendent-ai-labs.github.io/DynaML/core/core_ffn_new/.*

[46] Pierre Guillou. Understand how works Resnet… without talking about residual. *https://medium.com/@pierre_guillou/understand-how-works-resnet-without-talking-about-residual-64698f157e0c*, 2018.

[47] Vincent Fung. An Overview of ResNet and its Variants. *https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035?gi=70a1df0cbf5*, 2017.

[48] Shane Lynn. Get Busy with Word Embeddings – An Introduction. *https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/.*

[49] Andrew Tulloch. Fast Randomized SVD. *https://research.fb.com/wp-content/uploads/2016/11/post00049_image0001.png*, 2014.

[50] Pablo Gimeno Jordán, Alfonso Ortega Giménez y Ignacio Viñals Bailo. Segmentación automática de audio con modelos basados en redes neuronales para entornos broadcast. *https://deposita.unizar.es/record/41985?ln=es#*, 2018.

[51] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *https://arxiv.org/abs/1412.6980*, 2014.

[52] Sebastian Ruder. An overview of gradient descent optimization algorithms. *https://arxiv.org/abs/1609.04747*, 2016.

[53] Navneet Dalal and Bill Triggs: Histograms of Oriented Gradients for Human Detection. *https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf*

[54] Chris McCormick: Word2Vec - The Skip-Gram Model. *http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/,* 2016.

[55] Redes neuronales convolucionales con TensorFlow. *https://relopezbriega.github.io/images/conv_layer.png*

[56] Pranoi Radhakrishnan: Image Captioning in Deep learning. *https://towardsdatascience.com/image-captioning-in-deep-learning-9cd23fb4d8d2*, 2017

[57] Google Cloud Video Intelligence. https://cloud.google.com/video-intelligence/

[58] Microsoft Azure: Video Indexer. https://docs.microsoft.com/es-es/azure/cognitive-services/video-indexer/video-indexer-overview

[59] RTVE: Radio Televisión Española. *http://www.rtve.es/*

[60] Wildlife Sample Video. *https://www.youtube.com/watch?v=D25W67swuHs*, 2010.

[61] Sample Videos. *https://www.sample-videos.com/*

[62] Nikon D7500: En busca de la perfección. *https://www.youtube.com/watch?v=hRQy9BSIU0Q*, 2017.

[63] 24 horas: Edición mediodía. *https://www.youtube.com/watch?v=jbqZkJiPk7g*, 2015.

[64] Samsung Galaxy S7: Video sample. *https://www.youtube.com/watch?v=R5TfhaRBtsY*, 2016.