



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

**Generación de relatos a partir de una secuencia de imágenes
utilizando técnicas de aprendizaje profundo**

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

MAESTRA EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

P R E S E N T A

DIANA VIRGINIA GONZÁLEZ RICO

Director de Tesis:

Dr. Gibrán Fuentes Pineda

INSTITUTO DE INVESTIGACIONES EN

MATEMÁTICAS APLICADAS Y EN SISTEMAS

Ciudad Universitaria, CDMX

Diciembre, 2018



Universidad Nacional
Autónoma de México

Dirección General de Bibliotecas de la UNAM

Biblioteca Central



UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Resumen

La generación de relatos a partir de una secuencia de imágenes es un problema reciente en la comunidad de aprendizaje profundo. Este problema consiste en generar un relato como el que los seres humanos producen al hablar sobre un álbum de imágenes, en donde se expresa sobre los eventos, experiencias o ideas abstractas que pueden ser extraídas a partir de las imágenes.

En este trabajo de tesis se presenta un método basado en redes neuronales para la generación de relatos a partir de una secuencia de imágenes. Este método extiende el modelo *Show and Tell* [60] para descripción de imágenes utilizando una red recurrente, esta red codifica la secuencia de imágenes en un vector de contexto que representa la esencia de la secuencia. Posteriormente, se inicializan múltiples decodificadores con el vector de contexto para generar porciones del relato completo. Cada decodificador genera la porción correspondiente a la imagen que toma como primer entrada.

Los múltiples decodificadores utilizan información global e individual para generar el relato. Estos decodificadores independientes permitieron obtener modelos de lenguaje específicos para la posición en el relato.

Los relatos generados por el método propuesto obtuvieron resultados competitivos en las evaluaciones automática y humana de la competencia *Visual Storytelling Challenge* llevada a cabo como parte del *Storytelling Workshop*, en el marco del congreso del 2018 de la *North American Chapter of the Association for Computational Linguistics*.

Agradecimientos

A mi padre y madre, por siempre apoyarme para seguir mis sueños. Los sacrificios que han realizado me han permitido llegar hasta aquí; hoy, mañana y siempre, mis logros son suyos.

A mi hermana, por el cariño, apoyo incondicional y no dudar de mí. Por siempre estar, brindándome consejos y palabras de aliento.

A mis amigos que me acompañaron en este camino, nuestras pláticas llenas de risas hicieron mucho más placentera esta experiencia. Fue un gusto aprender con y de ustedes.

A mi mentor, el Dr. Gibrán Fuentes Pineda que me brindó una excelente guía durante el desarrollo de este trabajo. Por los conocimientos invaluable que me compartió y por ser una fuente de inspiración y admiración. Finalmente, por motivarme y confiar en mí.

A la UNAM, el Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas y el Posgrado en Ciencia e Ingeniería de la Computación por haberme brindado la oportunidad de continuar con mis estudios.

Al Consejo Nacional de Ciencia y Tecnología por la beca proporcionada durante este posgrado.

Índice general

Resumen	1
Agradecimientos	2
Índice de figuras	6
Índice de cuadros	8
1. Introducción	11
1.1. Motivación	14
1.2. Objetivos	15
1.2.1. Objetivo general	15
1.2.2. Objetivos particulares	15
1.3. Contribución de la tesis	16
1.4. Estructura de la tesis	16
2. Trabajo relacionado	18
2.1. Generación de descripción de imágenes	18
2.1.1. En imágenes aisladas	19
2.1.2. En imágenes en secuencia y video	20
2.2. Generación de historias	22
2.2.1. A partir de texto	22
2.2.2. A partir de imágenes	22

2.3. Generación de relatos a partir de una secuencia de imágenes	26
3. Marco teórico	29
3.1. Aprendizaje automático	29
3.1.1. Aprendizaje supervisado	31
3.2. Aprendizaje profundo	33
3.3. Redes neuronales	35
3.3.1. Optimización	38
Descenso por gradiente	38
Descenso por gradiente estocástico	41
3.3.2. Retropropagación	42
3.4. Redes neuronales convolucionales	44
3.4.1. Capas convolucionales	45
3.4.2. Capas de submuestreo	47
3.4.3. Capas completamente conectadas	48
3.4.4. Capas adicionales	48
3.5. Redes neuronales recurrentes	48
3.5.1. <i>Long Short Term Memory</i>	50
Compuerta de olvido	51
Compuerta de entrada	51
Compuerta de salida	52
3.5.2. Entropía cruzada	53
3.5.3. Retropropagación a través del tiempo	53
3.6. Modelos secuencia a secuencia	54
4. Generando relatos	56
4.1. <i>Show and Tell</i>	56
4.2. Componentes principales	58
4.2.1. <i>Inception V3</i>	58
4.2.2. Extrayendo la esencia	62

4.2.3. Generando un relato	64
4.3. Método propuesto	65
4.3.1. Fase de entrenamiento	66
4.3.2. Fase de inferencia	68
<i>Beam Search</i>	69
5. Experimentación y evaluación	70
5.1. Conjunto de datos	70
5.1.1. Descripción	72
5.1.2. Recopilación	72
5.2. Métricas de evaluación	74
5.2.1. BLEU	74
5.2.2. METEOR	75
5.2.3. CIDEr	76
5.2.4. ROUGE-L	77
5.3. Entrenamiento del método	78
5.3.1. Perplejidad en modelos de lenguaje	79
5.4. Evaluación del método	80
5.4.1. <i>Visual Storytelling Challenge</i>	80
Evaluación automática	81
Evaluación humana	81
5.4.2. Evaluación con métricas	83
5.5. Discusión	83
6. Conclusiones y trabajo futuro	86
Bibliografía	89
Anexo A. Ejemplos de relatos generados	97

Índice de figuras

1.1. Ejemplo de un relato generado a partir de una secuencia de imágenes. Reproducido de [54]	12
2.1. Ejemplos de descripción de imágenes aisladas. Reproducidos de [35, 60]	19
2.2. Ejemplos de descripción de imágenes en secuencia. Reproducidos de [19, 12]	21
2.3. Ejemplo de imagen de fondo utilizada en <i>Picture Books 2</i> . Reproducido de [2]	23
2.4. Ejemplos de historias generadas por <i>Neural-Storyteller</i> . Reproducidos de [48]	25
2.5. Ejemplo de un relato generado a partir de una secuencia de imágenes. Reproducido de [54]	27
2.6. Ejemplo de resumen y generación de relato de álbumes por los sistemas presentados en [64]. Reproducido de [64]	28
3.1. Arquitectura de una red neuronal profunda y una no profunda	34
3.2. Arquitectura general de una red neuronal	36
3.3. Ejemplo de cómo el descenso por gradiente utiliza la derivada de una función para encontrar la mejor solución posible	39
3.4. Tipos de puntos críticos en una función	40
3.5. Intuición del proceso de retropropagación de errores en una red neuronal	43
3.6. Arquitectura de una red neuronal convolucional	45

3.7. Ejemplo de operación de convolución en una red convolucional	46
3.8. Arquitectura general de una red recurrente	49
3.9. Arquitectura general de un modelo secuencia a secuencia	55
4.1. Arquitectura de <i>Show and Tell</i>	57
4.2. Arquitectura de la red <i>Inception V3</i>	59
4.3. Detalle de módulo inception utilizado en la arquitectura de <i>Inception V3</i> en donde cada convolución de 5×5 es reemplazada por una convolución de 3×3 . Reproducido de [53]	60
4.4. Detalle de módulo <i>inception</i> utilizado en la arquitectura de <i>Inception V3</i> después de la factorización de una convolución de $n \times n$, donde $n = 7$. Reproducido de [53]	61
4.5. Detalle de módulo <i>inception</i> utilizado en la arquitectura de <i>Inception V3</i> después de la factorización de una convolución de $n \times n$, donde $n = 8$. Reproducido de [53]	61
4.6. Codificador del método propuesto	63
4.7. Decodificador del método propuesto	64
4.8. Arquitectura del método propuesto	67
5.1. Gráfica de perplejidad en los datos de entrenamiento y validación de los experimentos del método propuesto	79
5.2. Ejemplos de relatos generados por el método propuesto	85

Índice de cuadros

1.1. Diferencia entre descripciones aisladas, descripciones en secuencia y generación de relatos. Imágenes y texto reproducidos de [54]	13
4.1. Detalle de las capas en <i>Inception V3</i> . Reproducido de [53]	60
5.1. Ejemplo de imágenes y texto en la base de datos VIST	71
5.2. Cantidad de imágenes en VIST originalmente y después de realizar filtrado	73
5.3. Cantidad de relatos en VIST originalmente y después de realizar filtrado	73
5.4. Resultados de la evaluación con METEOR de los relatos generados por el método propuesto	81
5.5. Evaluación humana de los relatos generados por los equipos de la competencia comparados contra relatos generados por humanos. Los aspectos evaluados son: a) ¿el relato es enfocado?, b) estructura y coherencia del relato, c) ¿compartirías el relato?, d) ¿el relato fue escrito por un humano?, e) ¿el relato está relacionado con las imágenes?, f) ¿qué tan detallado es el relato?	82
5.6. Evaluación con métricas automáticas de los relatos generados por el método propuesto en el conjunto de prueba de VIST	83

Notación

A continuación se presenta la notación matemática utilizada a lo largo de este trabajo. Esta notación está basada en la utilizada por Goodfellow et al. en su libro *Deep Learning* [17].

a	Un escalar (entero o real)
\mathbf{a}	Un vector
A	Una matriz
\mathbf{A}	Un tensor
\mathbb{A}	Un conjunto
\mathbb{R}	El conjunto de los números reales
$\{0, 1, \dots, n\}$	El conjunto de todos los enteros entre 0 y n
$[a, b]$	El intervalo real que incluye a a y b
\mathbf{a}_i	Elemento i del vector \mathbf{a} , con índices que inician en 1
$\frac{dy}{dx}$	Derivada de y con respecto a x
$\frac{\partial y}{\partial x}$	Derivada parcial de y con respecto a x
$\nabla_{\mathbf{x}} y$	Gradiente de y con respecto a \mathbf{x}

$\frac{\partial f}{\partial \mathbf{x}}$	Matriz Jacobiana $J \in \mathbb{R}^{m \times n}$ de $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$
$f : \mathbb{A} \rightarrow \mathbb{B}$	La función f con dominio \mathbb{A} y rango \mathbb{B}
$f(\mathbf{x}; \theta)$	Una función de \mathbf{x} parametrizada por θ . (Se suele escribir $f(\mathbf{x})$ y omitir el argumento θ para aligerar la notación)
$\log x$	Logaritmo natural de x
\mathbb{X}	Un conjunto de ejemplos de entrenamiento
$\mathbf{x}^{(i)}$	El i -ésimo ejemplo (de entrada) de un conjunto de datos
$y^{(i)}$	El objetivo asociado con $\mathbf{x}^{(i)}$ en aprendizaje supervisado
w_{ij}^k	El peso del nodo j en la capa l_k proveniente del nodo i
b_i^k	El sesgo para el nodo i de la capa l_k

1

Introducción

Una forma vital de comunicación para el ser humano es la comunicación visual. Todo lo que podemos apreciar mediante el sentido de la vista forma parte de nuestras experiencias personales. Nuestro aprendizaje, nuestras decisiones y nuestra personalidad se ven influenciados por las cosas que vemos todos los días. En la actualidad, vivimos rodeados de contenido visual que nos proporciona información: imágenes, videos, pinturas y esculturas, las cuales muchas veces pasan desapercibidas. Con un poco de atención, es posible apreciar que el mundo en el que habitamos es altamente visual.

Por otro lado, el contar historias es una parte fundamental de nuestra naturaleza humana. Gracias a esta aptitud lingüística, el ser humano percibe su realidad de una manera narrativa; el mundo mismo es un conjunto de relatos que cuentan nuestras vidas con el paso del tiempo. Con esta aptitud tan poderosa el hombre ha encontrado una forma de transmitir emociones y creencias. Es claro entonces, que tanto las imágenes como el lenguaje natural son un pilar importante de la experiencia humana y debido al poder que tienen las imágenes para contar historias, es natural pretender utilizarlas para generar relatos.

La generación de texto a partir de imágenes ha sido un problema muy popular en las comunidades de aprendizaje automático, procesamiento de lenguaje natural



“James and I were excited to be in Washington D.C. during the 4th of July. There was a huge crowd of people already awaiting the firework show. We were lucky to find a nice spot on the grass to watch the show. As the evening grew darker the crowd was gearing up to enjoy the show, with a great view of the Washington Monument. I was able to capture a great photo of the grand finale of the firework show.”

Figura 1.1: Ejemplo de un relato generado a partir de una secuencia de imágenes. Imágenes y relato reproducidos de [54].

y visión computacional, incluyendo problemas como la descripción de imágenes o video, la generación de historias y la resolución de preguntas acerca de éstas, por nombrar algunos. En los últimos años se ha generado un interés muy grande por la investigación en sistemas que a partir de una fuente visual construyen escritos en lenguaje natural [31, 45, 54, 60]. En particular, los avances recientes en estas áreas de investigación han permitido el desarrollo de sistemas capaces de describir imágenes cada vez con más precisión.

Del éxito en tareas como descripción de imágenes y descripción de video, surge el deseo de dar otro paso hacia adelante y resolver la tarea de la generación de relatos a partir de imágenes. La comunidad de inteligencia artificial ha trabajado en el desarrollo de sistemas generadores de narrativas. Sin embargo, sólo algunos de éstos lo hacen a partir de imágenes, y más bien parten de fuentes textuales o de un conjunto de palabras clave. Asimismo, aquellos sistemas que utilizan imágenes para generar texto están concentrados en simplemente generar descripciones de las imágenes o generar narraciones las cuales pueden ser consideradas desde cuentos hasta algún otro tipo de historia mucho más extensa.

Generar relatos a partir de imágenes es una tarea compleja y hasta este momento

				
<i>"A black frisbee is sitting on top of a roof."</i>	<i>"A man playing soccer outside of a white house with a red door."</i>	<i>"The boy is throwing a soccer ball by the red door."</i>	<i>"A soccer ball is over a roof by a frisbee in a rain gutter."</i>	<i>"Two balls and a frisbee are on top of a roof."</i>
<i>"A roof top with a black frisbee laying on the top of the edge of it."</i>	<i>"A man is standing in the grass in front of the house kicking a soccer ball."</i>	<i>"A man is in the front of the house throwing a soccer ball up."</i>	<i>"A blue and white soccer ball and black frisbee are on the edge of the roof top."</i>	<i>"Two soccer balls and a frisbee are sitting on top of the roof top."</i>
<i>"A discuss got stuck up on the roof."</i>	<i>"Why not try getting it down with a soccer ball?"</i>	<i>"Up the soccer ball goes."</i>	<i>"It didn't work so we tried a volley ball."</i>	<i>"Now the discuss, soccer ball, and volley ball are all stuck on the roof."</i>

Cuadro 1.1: Diferencia entre descripciones aisladas (primer renglón), descripciones en secuencia (segundo renglón) y generación de relatos (tercer renglón). Imágenes y texto reproducidos de [54].

son pocos los trabajos que intentan resolver este problema. En esta tarea, se intenta simular la capacidad de generar relatos a partir de álbumes de fotografías, como los relatos que los seres humanos proveen al hablar sobre un álbum de imágenes, como solemos hacerlo en nuestra vida diaria. En la figura 1.1 se muestra un ejemplo de un relato generado a partir de una secuencia de imágenes. Asimismo, en el cuadro 1.1 se presenta una comparación entre las tareas de descripción de imágenes aisladas y en secuencia y generación de relatos a partir de imágenes.

1.1. Motivación

Un sistema capaz de crear relatos a partir de una secuencia de imágenes involucra que no solamente pueda detectar los objetos, atributos y las acciones que se encuentran en cada imagen, sino que pueda enlazar eficazmente las escenas independientes de eventos a través del tiempo y encontrar las relaciones más relevantes para poder generar una narración cohesiva. Por lo tanto, esto demanda pasar de un razonamiento de imágenes aisladas, en donde se trabaja con escenas estáticas y no se necesita un contexto, a un razonamiento narrativo en donde una secuencia de imágenes representa ciertos eventos los cuales evolucionan con el tiempo.

Los sistemas que describen imágenes son utilizados por ejemplo en aplicaciones que buscan describir detalladamente escenas para personas discapacitadas visualmente. Un sistema que hable sobre los objetos y eventos en las imágenes de forma más conversacional, puede describirlas con una sensación de naturalidad y no tan robótica.

Con el avance de la tecnología y el crecimiento de su presencia en nuestras vidas, las imágenes se han vuelto un medio aún más importante con el cual nos es posible transmitir información y compartir experiencias. Aunado a esto, el aumento de usuarios en las redes sociales ha dado lugar a que los usuarios publiquen álbumes de fotografías para compartir sus experiencias y usualmente, estas imágenes son acompañadas de comentarios escritos por los mismos usuarios que explican o hablan de los momentos capturados en ellas.

Por otro lado, la generación automática de relatos a partir de una secuencia de imágenes podría servir como base para la creación de distintos tipos de narraciones, dando paso a sistemas que a partir de una secuencia de imágenes sean capaces de generar chistes, poemas, cuentos para niños o historietas. Adicionalmente, con el avance de la tecnología y el aumento de usuarios en las redes sociales, se ha vuelto más común que las personas deseen compartir experiencias mediante fotografías y publicar dichas imágenes con una descripción o comentario sobre ellas. Un sistema

que realice un relato de las experiencias en estos conjuntos puede ayudar a automatizar tales comentarios y de esta manera ahorrar tiempo a los usuarios.

1.2. Objetivos

1.2.1. Objetivo general

El objetivo general de este trabajo de investigación es desarrollar un método computacional que sea capaz de generar relatos a partir de una secuencia ordenada de imágenes. Estos relatos deben hablar de forma natural sobre el contenido y eventos en las imágenes.

1.2.2. Objetivos particulares

- Diseñar una arquitectura de aprendizaje profundo que genere relatos a partir de una secuencia de 5 fotografías reales. Estas fotografías constituyen una secuencia de imágenes ordenadas y que de alguna manera cuentan una historia, tales como las que pueden ser encontradas en redes sociales o álbumes familiares.
- Evaluar los relatos generados por el método propuesto de manera cuantitativa haciendo uso de métricas provenientes de tareas más ampliamente evaluadas como la traducción automática y la descripción de imágenes.
- Evaluar los relatos generados por el método propuesto de manera cualitativa mediante una evaluación humana que considere los aspectos más importantes que deben contener los relatos, tales como coherencia, relación con las imágenes y corrección gramatical, por mencionar algunos.

1.3. Contribución de la tesis

En el presente trabajo de investigación se expone un nuevo método capaz de generar relatos a partir de una secuencia ordenada de imágenes. Este método se encuentra basado en arquitecturas de aprendizaje profundo, las cuales en conjunto son capaces de extraer información relevante de la secuencia de imágenes y utilizarla para generar relatos que incorporan los objetos y eventos presentes en las imágenes.

En específico, el método consiste en una arquitectura secuencia a secuencia, en donde se tiene un codificador que resume el contenido de las imágenes de la secuencia y varios decodificadores en donde cada uno se encarga de generar una frase que forma parte del relato final. Con el uso de decodificadores independientes se obtienen modelos de lenguaje más específicos para cada posición en el relato. Asimismo, cada decodificador es alimentado con información particular y global que lo ayuda a generar relatos más informados sobre el contenido de las fotografías. Finalmente, esta arquitectura genera relatos coherentes y gramaticalmente correctos en donde se describen las escenas de las fotografías de una manera más conversacional.

1.4. Estructura de la tesis

Esta tesis se encuentra conformada de la siguiente manera:

- **Capítulo 2** Se encuentra dividido en dos secciones. En la primer sección se abordan brevemente otras formas de generación de texto a partir de imágenes que se relacionan con la generación de relatos. La segunda sección describe el problema de generación de relatos a partir de imágenes, así como los trabajos más representativos existentes hasta el momento.
- **Capítulo 3** En este capítulo se proporciona información relevante sobre el área de aprendizaje automático y aprendizaje profundo. Después se presentan conceptos y arquitecturas importantes de aprendizaje profundo de las cuales se

hace uso en el método presentado en este trabajo.

- **Capítulo 4** Se explica en detalle el método propuesto para la generación de relatos a partir de una secuencia de imágenes. Se discuten sus componentes y su funcionamiento durante las fases de entrenamiento e inferencia.
- **Capítulo 5** En este capítulo, se comienza explicando el conjunto de datos que fue utilizado para el entrenamiento de la arquitectura propuesta y se describen las métricas automáticas para la evaluación de los relatos generados. Se presentan los resultados obtenidos con el modelo y su evaluación con las distintas métricas propuestas. Asimismo, se presenta el desempeño que tuvo el modelo al ser sometido a una competencia de generación de relatos a partir de imágenes. Finalmente, se realiza una discusión sobre el desempeño del modelo en general.
- **Capítulo 6** Se presentan las conclusiones del trabajo así como posibles mejoras al método propuesto para aumentar la calidad de los relatos generados.

2

Trabajo relacionado

Debido a lo compleja y reciente que es la tarea de generación de relatos a partir de imágenes, existen muy pocos trabajos en los que se intente resolverla. Sin embargo, la generación de relatos puede verse relacionada y ha tomado inspiración de otras tareas, las cuales también involucran la unión de visión computacional y lenguaje natural.

2.1. Generación de descripción de imágenes

Una de las tareas muy relacionada con la generación de relatos es la descripción de imágenes. Esta tarea se ha convertido en uno de los problemas más populares en la comunidad de aprendizaje automático debido a la considerable cantidad de esfuerzos científicos recientes dedicados a resolverla. [30, 27, 55, 60]

Esta tarea requiere de un proceso capaz de detectar tanto los objetos como sus atributos en una imagen el cual es un problema bien estudiado en el área de visión computacional. Los avances que se han realizado han permitido describir imágenes cada vez con más precisión [10, 55, 60].



"This is the picture of two dogs. The first dog is near the second furry dog."

"A little girl that is sitting on a bed."

Figura 2.1: Ejemplo de descripción de imágenes aisladas. Reproducidos de [35] en la izquierda y [60] en la derecha.

2.1.1. En imágenes aisladas

Los trabajos que se mencionarán en esta sección intentan generar la descripción de una sola imagen. Para que un sistema pueda describir una imagen adecuadamente necesita información sobre objetos, atributos y relaciones, así como la capacidad de enlazar todo esto en una descripción fluida y coherente en un lenguaje comprensible para los seres humanos. Los sistemas más antiguos se basaron en modelos como grafos *And-Or* en la parte visual y sistemas basados en reglas para la generación de la descripción. Trabajos posteriores [13, 35] utilizaron tripletas de información (e.g. tripletas del tipo $\langle \text{objeto}, \text{acción}, \text{escena} \rangle$) para generar las descripciones pero estas resultaron ser lingüísticamente pobres. En la figura 2.1 se presenta un ejemplo de descripción generada por el sistema de Kulkarni et al. [35].

Los trabajos de [30, 40, 41] se basan en predecir la siguiente palabra de la descripción dadas las palabras anteriores y la imagen. En el trabajo de Kiros et al. [30] se utiliza una red de propagación hacia delante y en [41, 40] utilizan redes recurrentes para generar la descripción. Más recientemente, en el trabajo de [27] se presenta

un alineamiento entre regiones y palabras que permite crear las descripciones. Vinyals et al. [60] propone una arquitectura conocida como *Show and tell* parecida a la de [30, 41, 40] con la diferencia de utilizar lo que ellos denominan como un modelo de lenguaje más poderoso alimentado con información visual directamente que ayudó a tener mejores resultados. La arquitectura de este método fue utilizada como base para este trabajo y será explicada en detalle más adelante. En la figura 2.1 se presenta un ejemplo de descripción generada por este sistema.

2.1.2. En imágenes en secuencia y video

La tarea de realizar descripciones a partir de una secuencia de imágenes se concentra en que la descripción de las imágenes se ve influenciada no solamente por las características que pueden observarse en la imagen actual, sino también por el contenido de la(s) imagen(es) anterior(es). Este problema es muy cercano a la descripción de videos, en donde los videos son representados mediante una secuencia de marcos.

Al igual que en la descripción de imágenes aisladas, existe abundante bibliografía sobre la descripción de video. El trabajo de [59] presenta un modelo secuencial que se utiliza para generar descripciones de video, el cual utiliza una red convolucional para la representación de la imagen y una red recurrente para generar la descripción en lenguaje natural. Más recientemente, Guo et al. [19] utilizó modelos de descripción de imágenes aisladas para generar una descripción de cada una de las imágenes de la secuencia y después unificar todas las descripciones en una sola, generando la descripción final del video. El modelo propuesto por [12] involucra redes convolucionales y una red recurrente LSTM de dos capas. La representación de la imagen obtenida por la red convolucional se utiliza por la red recurrente para generar las palabras en la descripción. A diferencia del trabajo de Venugopalan et al. [59], su sistema es capaz de generar no una sino un conjunto de oraciones que describen la secuencia, y a partir de ellas, elegir aquella que mantiene una relación más estrecha con la secuencia, evaluándolas mediante una función de energía.



"I am talking with a friend while eating a meal in a restaurant."



"A woman is slicing a cucumber."

Figura 2.2: Ejemplos de descripción de imágenes en secuencia. Arriba se muestra un ejemplo de las descripciones generadas por el sistema de [12] y abajo un ejemplo de las generadas por el sistema de [19].

Otro trabajo sobresaliente para la generación de varias oraciones que describen una secuencia de imágenes es el de Park y Kim [45]. Su modelo es entrenado con dos conjuntos de datos (*NYC* y *Disney*), los cuales son una colección de publicaciones pertenecientes a blogs redactados por personas sobre sus experiencias en la ciudad de Nueva York o el parque de atracciones *Disneyland*. El modelo presentado consiste en redes convolucionales para la representación de las imágenes, redes recurrentes bidireccionales para la generación de las oraciones y un modelo de coherencia que permite unir las oraciones de forma más natural.

2.2. Generación de historias

2.2.1. A partir de texto

La generación de narraciones ha sido un problema de mucho interés en la inteligencia artificial desde hace muchos años. Un ejemplo de esto es el sistema generador de narraciones *Novel Writer* [32]. Este sistema creaba historias que giraban en torno a un misterio de asesinato en una fiesta. Siendo este el primer sistema de este tipo del que se tiene registro.

Algunos de los trabajos realizados en esta sección conciernen sistemas basados en reglas, los cuales utilizan una base de conocimientos y, mediante reglas y algunas otras relaciones que son definidas por el programador, se generan historias relacionadas a algún tema en específico. Entre ellos se encuentra el trabajo de Pérez y Pérez [46] el cual ha sido utilizado como base por muchos otros trabajos. Su sistema MEXICA define un marco en el que un agente genera la trama de una historia relacionada a los mexicas. A partir de este trabajo se han tenido múltiples variantes que intentan enfoques diferentes para una mejora de los resultados del sistema, como en [15] en donde se combinan otras áreas como lo son los algoritmos evolutivos. Una variante más reciente es el sistema MABLE [50] que está basado en el sistema MEXICA y un modelo estadístico, el cual expande la historia generada por MEXICA convirtiéndola en una balada mediante la inserción de rimas coherentes en la historia.

2.2.2. A partir de imágenes

Una cantidad menor de los trabajos realizados sobre generación de narraciones utilizan imágenes. Entre estos trabajos se encuentra el sistema *Picture Books* [22] que explora técnicas de generación de lenguaje natural con el objetivo de crear historias para niños de cuatro a seis años. A partir de una imagen seleccionada por el usuario, el sistema es capaz de crear historias coherentes las cuales contienen palabras

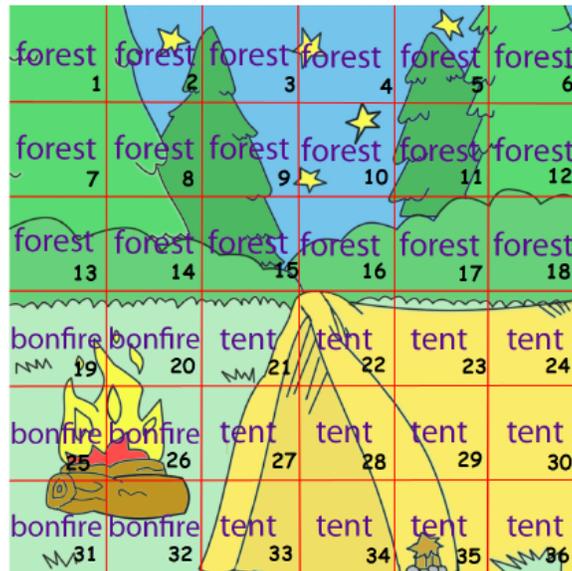


Figura 2.3: Ejemplo de imagen de fondo utilizada en *Picture Books 2*. Reproducido de [2].

y estructuras gramaticales apropiadas para niños; las historias generadas son consideradas fábulas pues reflejan lecciones morales en la trama. El diseño del sistema intenta modelar la manera en la que los seres humanos cuentan historias, tomando en cuenta los objetivos de los personajes.

Picture Books 2 [2] es una extensión de [22], en donde explican la importancia de utilizar una secuencia de imágenes para crear historias; dado que una historia es una secuencia de eventos o escenas y el uso de una sola imagen limita el contenido en la narración generada. Este sistema cuenta con una interfaz que permite a niños de seis a ocho años de edad definir múltiples imágenes como entrada para que el sistema genere una historia con lección moral. Se conforma de cuatro módulos principales: representación de la escena, planificador de la historia, planificador de oraciones y generación de la historia. El primer módulo permite al niño elegir ciertos componentes de una lista predefinida, como lo son la imagen de fondo para la historia (como la mostrada en la figura 2.3), los personajes y objetos a utilizar. Asimismo, una imagen de entrada para el sistema que contenga mínimo tres escenas que permitan repre-

sentar la historia. Después, una representación abstracta de la imagen de entrada es generada por el planificador de la historia mientras que el planificador de oraciones se encarga de lexicalizar conceptos y generar un conjunto de especificaciones para las oraciones a utilizar. Finalmente, la historia es creada mediante el último módulo haciendo uso de *simpleNLG* [58].

Dado que ambos sistemas anteriormente mencionados tienen el objetivo de generar historias para niños, las imágenes utilizadas representan dibujos como el mostrado en la figura 2.3, los cuales pueden parecer más atractivos y adecuados para niños menores de ocho años.

Uno de los trabajos más cercanamente relacionados con el presentado en esta investigación es el de Kiros et al. [31]. Su modelo de arquitectura secuencia a secuencia intenta generar las oraciones alrededor de una oración de entrada. Inspirado en este trabajo, se encuentra el sistema *Neural-Storyteller*, basado en redes neuronales recurrentes y vectores de pensamiento presentados en [31]. Asimismo, hace uso de cambios de estilo, una operación que permite al modelo transferir estilos convirtiendo descripciones estándar de imágenes a un tipo de historia como el encontrado en novelas.

El sistema cuenta con una red neuronal recurrente que fue entrenada sobre novelas románticas. Cada novela procesada es codificada en un vector de pensamiento el cual posteriormente es decodificado por la red para generar el pasaje de dicha novela. La red recurrente es entrenada sobre un subconjunto de historias románticas del conjunto de datos *BookCorpus* [65]. De forma paralela, se aprende una representación entre imágenes y descripciones tomadas del conjunto de datos MSCOCO [38], donde las imágenes y las descripciones son mapeadas a un mismo espacio vectorial. Este marco les permite generar historias a partir de imágenes pues los vectores de pensamiento ofrecen una forma de mapear descripciones de imágenes a pasajes de libros. En la figura 2.4 se muestran ejemplos de las historias generadas por su sistema.



"I had a man in control, and Kate turned her head to look at him. There was no denying it, no matter what she said. In the past few days, I couldn't help but notice how beautiful the band was. The man who stood at the end of the hall told me he was in love with her, and that's the only way I would ever be able to get my emotions out of my head. Her lips curled into a tight little O, as if waiting for an audience. He seemed to remember something more than a year ago, the man who and I was breaking down."



"I give you a man, I don't know what's happening to me, and when I look back at the stage, I say, God, I love you more than I should."

Figura 2.4: Ejemplos de historias generadas por *Neural-Storyteller*. Imágenes e historias reproducidas de [48].

2.3. Generación de relatos a partir de una secuencia de imágenes

En la sección anterior se habló acerca de trabajos que generan textos de longitud extensa que cuentan alguna historia, algunos de ellos a partir de imágenes. Sin embargo, el problema en el que se enfoca este trabajo de investigación toma inspiración de ellos pero no intenta imitarlos.

La generación de relatos en el contexto de este trabajo es la generación de un texto corto a partir de una secuencia de imágenes cuyo contenido se refiere a los eventos y objetos en las imágenes intentando imitar la forma en que un ser humano lo haría. Entre los pocos trabajos que intentan resolver esta tarea, se encuentra el de Liu et al. [39] en donde se presenta un modelo basado en redes recurrentes bidireccionales. Lo que lo diferencia de los trabajos anteriores es el uso de sGRU (*skip Gated Recurrent Units*), estas permiten el alineamiento de historias locales en una secuencia completa. En su momento fue el primer trabajo relacionado con resolver la tarea de generar relatos para un conjunto de imágenes.

Entre los trabajos más recientes, se encuentra el de Huang et al. [54], en donde se presenta un primer conjunto de datos creado específicamente para la tarea que nos concierne y del cual se hace uso en este trabajo. Dicho conjunto de datos denominado SIND (*Sequential Image Narrative Dataset*) o VIST (*Visual Storytelling Dataset*), como es llamado actualmente, consiste en un conjunto de imágenes agrupadas en varias secuencias utilizadas para la generación de relatos que contengan una historia relacionada al contenido de las imágenes.

Los autores explican que la existencia de este tipo de conjuntos de datos es esencial para el desarrollo de sistemas basados en redes neuronales que puedan generar relatos más similares a las creadas por los humanos. Asimismo, proponen un modelo base para la solución del problema que consiste en una arquitectura codificador decodificador utilizando redes recurrentes.



“The family got together for a cookout. They had a lot of delicious food. The dog was happy to be there. They had a great time on the beach. They even had a swim in the water.”

Figura 2.5: Ejemplo de un relato generado a partir de una secuencia de imágenes utilizando el conjunto de datos VIST. Imagen y relato reproducidos de [54].

Dado que los resultados obtenidos con este modelo fueron austeros en la riqueza del lenguaje y repetitivos, los autores intentaron aumentar la calidad de los relatos generados haciendo uso de heurísticas simples tales como reducir el tamaño de la búsqueda y no permitir que el mismo contenido sea repetido dentro de la misma historia, entre otras.

En el trabajo de Agrawal et al. [1], se presentan varios modelos para la generación de secuencias de imágenes con relatos usando la base de datos VIST. Es importante destacar que ellos no generan los relatos como tal, sino que se enfocan en armar una historia tomando varios pequeños relatos desacomodados de un conjunto de datos y acomodarlos de tal manera que se construya una historia final coherente. El objetivo de su trabajo es resolver la tarea de generación de secuencias y para esto se exploraron varios modelos que fueron capaces de extraer un sentido común temporal, el cual les permitió enlazar historias coherentemente.

Recientemente, Yu et al. [64] presentan un modelo capaz de realizar resúmenes de álbumes de fotografías y generación de relatos sobre el conjunto de datos VIST. Su arquitectura consiste en tres redes recurrentes con atención, donde la primera red es el codificador del álbum, la segunda el selector de fotos y finalmente una tercera red que genera el relato. En la figura 2.6 se presenta esta arquitectura. El codificador del álbum es una red recurrente bidireccional GRU que en cada paso codifica una imagen

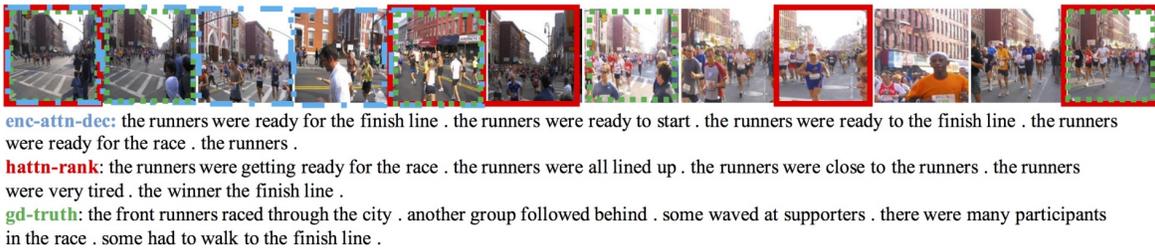


Figura 2.6: Ejemplo de resumen y generación de relato de álbumes por los sistemas presentados en [64]. Se presentan los resultados de dos sistemas (en azul y rojo) y los relatos de referencia (en verde). El codificador del modelo obtiene la esencia del álbum para que el selector de imágenes obtenga las imágenes más representativas (en recuadros de color) que conformarán la secuencia resumen del álbum. Finalmente, el decodificador genera el relato para la secuencia de imágenes seleccionadas. Reproducido de [64].

del álbum, obteniendo así, el contexto del álbum completo. Para la representación de las imágenes utilizan una red convolucional llamada *ResNet101* [20]. El selector de fotos es una red recurrente GRU que se encarga de generar una probabilidad de ser seleccionada para cada imagen del álbum. Aquellas cinco imágenes con mayor probabilidad, son las utilizadas para la generación del relato. Finalmente, el generador del relato utiliza la representación del álbum obtenida del codificador del álbum y las probabilidades generadas por el selector de fotos para generar el relato de la secuencia de las cinco imágenes seleccionadas. Su selector de imágenes les permitió lidiar con álbumes de diferentes cantidades de imágenes, a diferencia del modelo presentado por [54] el cual solamente realiza relatos sobre secuencias que contienen cinco imágenes. Sin embargo, un alto porcentaje de los relatos generados por su sistema pueden contener errores gramaticales y oraciones sin sentido.

3

Marco teórico

En este capítulo se presentan conocimientos necesarios sobre aprendizaje automatizado y aprendizaje profundo para una mejor comprensión de este trabajo. Para una instrucción más detallada sobre la materia se recomienda consultar el libro de Goodfellow et al. [17].

3.1. Aprendizaje automático

La necesidad de crear programas de computadora que pudieran aprender a partir de datos, dio lugar al surgimiento de una rama de la inteligencia artificial llamada aprendizaje automático. Una definición popular de lo que se intenta lograr con el aprendizaje automático es la provista por Tom Mitchell en [43]: *“Un programa de computadora se dice que aprende de cierta experiencia E , con respecto a una tarea T y una medida de rendimiento P , si su rendimiento en la tarea T , medido por P , mejora con experiencia E .”*

De la definición anterior, se abordarán por separado cada uno de sus elementos para esclarecer la esencia de los algoritmos de aprendizaje automático o “programas de computadora”. Se busca aprender a resolver un problema o tarea T que suele ser difícil de resolver mediante algoritmos convencionales diseñados por humanos.

Esta tarea muchas veces puede ser vista como un mapeo $f : \mathbb{X} \rightarrow \mathbb{Y}$ donde \mathbb{X} es el espacio de los datos de entrada (comúnmente conocido como “conjunto de datos de entrenamiento” o simplemente “datos de entrenamiento”) y \mathbb{Y} el espacio de los datos de salida.

La experiencia E es adquirida mediante un entrenamiento el cual consiste en procesar *ejemplos*. Un *ejemplo* puede ser visto como un conjunto de atributos que caracterizan el evento que se desea procesar. Este ejemplo es representado frecuentemente como un vector $a \in \mathbb{R}^d$ en donde cada elemento a_i es un atributo diferente. Adquirir esta experiencia para poder resolver la tarea es lo que se entiende como aprendizaje.

Una medida de rendimiento P es necesaria para poder evaluar si el algoritmo es capaz de resolver la tarea (o no), y qué tan bien la realiza. Debido a la gran variedad de tareas que pueden intentar resolverse, esta medida de rendimiento debe ser diseñada específicamente para la tarea en cuestión, de manera que sea una medida de desempeño apropiada. El objetivo principal es que al final del entrenamiento el algoritmo realice correctamente la tarea en datos que no ha visto antes. Es decir, aprenda a resolver la tarea a partir de un conjunto de datos conocido y logre generalizarla a datos desconocidos. Por esto, es importante que la medida de rendimiento P sea aplicada en un conjunto de datos conformado por ejemplos distintos a los que fueron utilizados para el entrenamiento del algoritmo. El entrenamiento y evaluación del algoritmo es usualmente realizado con tres distintos subconjuntos de los datos disponibles. Estos subconjuntos son entrenamiento, validación y prueba. El conjunto de datos de entrenamiento es usualmente el más grande y como su nombre lo indica, es utilizado para entrenar el algoritmo. El conjunto de validación es utilizado para una evaluación inicial y ajustar los hiperparámetros del algoritmo. Finalmente, el conjunto de prueba es utilizado para la evaluación del algoritmo una vez que el entrenamiento ha terminado y se ha realizado la evaluación inicial a través del conjunto de validación. Los conjuntos de validación y prueba suelen ser de aproximadamente el mismo tamaño.

El tipo de entrenamiento permite clasificar a los algoritmos de aprendizaje au-

tomático en aprendizaje supervisado, aprendizaje no supervisado o aprendizaje por refuerzo. El método presentado en este trabajo pertenece a la primer categoría, así que esta será explicada en detalle en la siguiente sección.

Entre los tipos de problemas que se han logrado resolver mediante aprendizaje automático se encuentran tareas de clasificación, regresión, transcripción, traducción y detección de anomalías, por nombrar algunas. En algunas de estas tareas puede ser difícil determinar qué medida de rendimiento utilizar, muchas veces porque no es sencillo decidir qué se debe evaluar.

Para aterrizar los elementos explicados anteriormente tomemos el siguiente ejemplo de un algoritmo de aprendizaje automático. Se desea resolver una tarea T que consiste en clasificar flores de iris en tres posibles categorías: *iris virginica*, *iris setosa* e *iris versicolor*. La experiencia E será obtenida mediante el procesamiento de los datos de entrada \mathbb{X} que corresponde a un conjunto de ejemplos donde cada uno consiste de un número determinado de atributos que lo caracterizan y los datos de salida \mathbb{Y} corresponde a únicamente las tres posibles categorías deseadas. Algunos de los atributos de cada ejemplo podrían ser el color de la flor, el largo del pétalo, el ancho del pétalo, etc. Como medida de rendimiento para esta tarea se suele medir la exactitud del programa la cual consiste en la proporción de ejemplos clasificados correctamente respecto a la cantidad total de ejemplos que se intentaron clasificar.

3.1.1. Aprendizaje supervisado

La desventaja al realizar un mapeo $f : \mathbb{X} \rightarrow \mathbb{Y}$ radica en que algunas veces puede ser difícil para el programador definir explícitamente f . En el ejemplo de flores iris, f plantearía el problema de especificarle al algoritmo cómo reconocer correctamente cada una de las tres categorías. Para tratar de solucionar este problema, en el aprendizaje supervisado se utiliza un conjunto de datos de entrenamiento etiquetados, es decir, en donde cada uno de los ejemplos cuenta con un conjunto de atributos y una etiqueta que representa la respuesta deseada; estas etiquetas son usualmente

anotadas por humanos.

Entonces en el aprendizaje supervisado los datos de entrenamiento de aquí en adelante denotados por \mathcal{D}_E es el conjunto de n ejemplos:

$$\mathcal{D}_E = \{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(n)}, y^{(n)})\} \quad (3.1)$$

que son procesados durante el entrenamiento, donde $\mathbf{x}^{(i)}$ corresponde a un vector de atributos y cada $y^{(i)}$ es la respuesta deseada del ejemplo i . Se asume que estos ejemplos son independientes e idénticamente distribuidos obtenidos de una distribución de datos \mathcal{D} .

Durante el entrenamiento se intenta aprender el mapeo $f \in \mathcal{F}$ a partir de un conjunto de funciones de las cuales se escoge aquella que sea más consistente con los datos de entrenamiento \mathcal{D}_E . En general, se intenta aprender f^* que minimice la pérdida esperada sobre la distribución de datos \mathcal{D} :

$$f^* = \arg \min_{f \in \mathcal{F}} \mathbf{E}_{(x,y) \sim \mathcal{D}} L(f(\mathbf{x}), y) \quad (3.2)$$

donde $L(f(\mathbf{x}), y)$ representa la función de pérdida o rendimiento y su valor es un escalar que básicamente mide la discrepancia entre la respuesta deseada y la respuesta predicha por el modelo.

La ecuación 3.2 se vuelve intratable debido al hecho de que no se tiene acceso a todos los posibles ejemplos en \mathcal{D} . Sin embargo, esto puede ser resuelto haciendo uso de la suposición anteriormente mencionada de que los datos en \mathcal{D}_E son independientes e idénticamente distribuidos, lo cual nos lleva a tener:

$$f^* \approx \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n L(f(\mathbf{x}^{(i)}), y^{(i)}) \quad (3.3)$$

es decir, se minimiza el error o pérdida solamente sobre los datos de entrenamiento \mathcal{D}_E , esperando sean un buen representante de la optimización sobre \mathcal{D} .

El objetivo del entrenamiento es que a partir de un conjunto de datos de entrenamiento \mathcal{D}_E , encontrar aquella función f^* que minimice el error entre lo que el modelo

predice y las respuestas deseadas en \mathcal{D}_E . Una vez encontrada y aprendida esta función se utilizará para mapear correctamente ejemplos nuevos.

Desafortunadamente, al tratar de resolver la ecuación 3.3 se presentan algunos problemas. Si la función encontrada f^* no es la apropiada, es posible que no sea capaz de generalizar bien a todos los datos $(x, y) \sim \mathcal{D}$. Por otra parte, se pueden tener múltiples f^* diferentes que logren tener el mismo error mínimo pero generalicen diferente sobre \mathcal{D} . Para resolver estos problemas se puede hacer uso de un término de regularización:

$$f^* \approx \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n L(f(\mathbf{x}^{(i)}), y^{(i)}) + R(f) \quad (3.4)$$

aquí, $R(f)$ representa la preferencia de algunas funciones sobre otras, independientemente de su adecuación a los datos de entrenamiento. Este término puede ser visto como una medida de complejidad de f^* y permite encontrar una f^* de baja complejidad que se adecúa bien a los datos de entrenamiento. Existen distintas maneras de regularizar, unas de las más populares es introducir la norma ℓ^2 o la norma ℓ^1 que obligan a que los parámetros de la red tomen valores más pequeños. Lo anterior surge de la teoría de que parámetros más pequeños conduce a modelos más simples.

3.2. Aprendizaje profundo

Aunque muchas tareas han sido resueltas exitosamente mediante algoritmos de aprendizaje automático, al tratar con problemas más complejos de la inteligencia artificial como reconocimiento de objetos o generación de texto, estos algoritmos no representan una técnica adecuada para su posible solución.

Una de las razones principales de que estos algoritmos no sean apropiados para problemas tan complejos se encuentra cuando se trabaja con datos de dimensionalidad alta. Es decir, si se tiene un conjunto de datos en los cuales se cuenta con n ejemplos y k atributos o características, si k es muy grande, el número de posibles

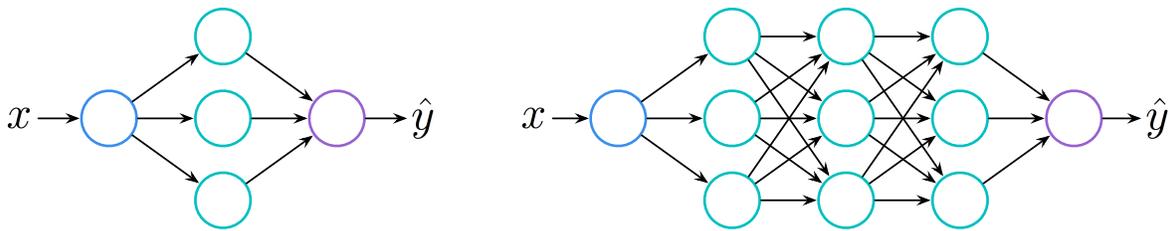


Figura 3.1: Se muestra la arquitectura de una red neuronal no profunda (izquierda) y una profunda (derecha), ambas con entrada x y salida \hat{y} . Una red neuronal tiene tres tipos de capas: capa de entrada (azul), capas ocultas (verde) y capa de salida (morado). La cantidad de capas ocultas determina la profundidad de la red.

combinaciones para encontrar una solución puede crecer rápidamente. Entonces, un conjunto de datos con dimensionalidad alta es aquél en donde el número de posibles combinaciones es muy grande con respecto a n . Esta desventaja, entre otras, dieron origen al nacimiento de una subárea del aprendizaje automático llamada **aprendizaje profundo**.

Uno de los modelos matemáticos más comunes hoy en día en el aprendizaje automático son las redes neuronales. Como modelo de aprendizaje automático, la tarea de una red neuronal es aproximar una función. Con un conjunto de datos de entrenamiento \mathcal{D}_E , la red neuronal es entrenada para estimar una función f . Una vez entrenada, dada una entrada x , la red realiza las operaciones necesarias mediante sus capas para devolver una salida \hat{y} . Aunque los detalles sobre las redes neuronales se presentan en la sección 3.3, la figura 3.1 presenta dos tipos de arquitecturas que permiten denotar la diferencia entre una red neuronal en aprendizaje automático y una en aprendizaje profundo.

Gracias a su gran cantidad de capas, las redes neuronales profundas permiten aprender funciones más complejas pero necesitan de mucha cantidad de datos para lograrlo. Es por esto que es preferible que las bases de datos utilizadas para entrenar redes profundas estén conformadas de muchísimos ejemplos pues esto puede ayudar

a obtener mejores resultados [51].

En las siguientes secciones se presenta una explicación sobre las redes neuronales y el método de optimización más común utilizado en estas, así como el proceso para realizarlo. Finalmente, se presenta una introducción a algunas arquitecturas de aprendizaje profundo que son cruciales para el método presentado en este trabajo.

3.3. Redes neuronales

En la sección anterior se mencionan brevemente las redes neuronales pero no se profundizó en ellas, así que a continuación se explica en detalle su arquitectura y funcionamiento.

Como se explicó en la sección 3.1.1, se busca aproximar una función f lo mejor posible. Esta aproximación f^* es representada mediante una **red neuronal** la cual consiste en un conjunto de neuronas conectadas.

Con inspiración en las neuronas de un cerebro humano y su funcionamiento, las redes neuronales son un modelo matemático computacional que representan composiciones de funciones, donde cada capa representa una función. Por ejemplo, si se tiene la siguiente composición $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$ entonces $f^{(1)}$ es representada por la primer capa de la red, $f^{(2)}$ es representada por la segunda capa y así sucesivamente. Una red neuronal consta de una primer capa llamada **capa de entrada**, una última capa llamada **capa de salida** y finalmente una o varias **capas ocultas** que se encuentran entre estas dos, como se muestra en la figura 3.1. Durante el entrenamiento de la red neuronal se busca que la red utilice sus capas para lograr aproximar la función f en cuestión lo mejor que pueda, guiándose con los datos que se encuentran en \mathcal{D}_E .

En la parte inferior de la figura 3.2 se presenta la arquitectura de una red neuronal de tres capas ocultas. Cada neurona de una misma capa está conectada con todas las neuronas de la siguiente capa pero nunca se conectan entre ellas. Este tipo de redes son llamadas redes de propagación hacia adelante debido a que todas las conexiones

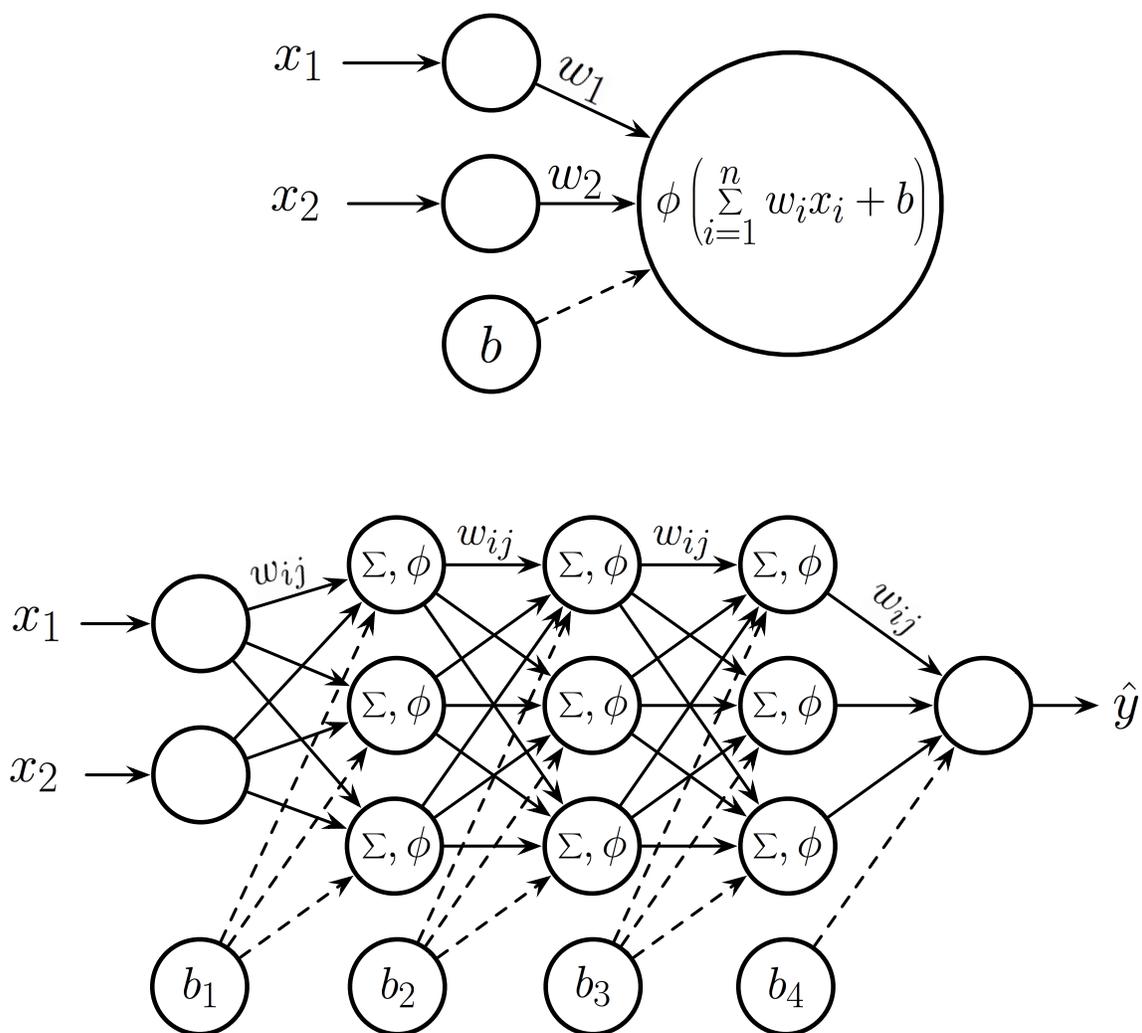


Figura 3.2: **Arriba** En detalle lo que sucede en todas las neuronas de las capas ocultas y la capa de salida. Se realiza la suma del producto de los pesos $\{w_1, w_2\}$ de todas las neuronas de la capa anterior $\{x_1, x_2\}$ que se conectan a la neurona actual. A este resultado se le suma un valor de sesgo b de la capa anterior. El resultado final es pasado por una función de activación ϕ . **Abajo** Arquitectura general de una red neuronal con tres capas ocultas. Las conexiones (flechas) van solamente hacia delante y cada una tiene un peso w_{ij} asociado que corresponde a la conexión entre la neurona i y la neurona j . Se tienen también b_1, \dots, b_{k-1} elementos de sesgo donde $k + 1$ es el número de capas en la red.

son solamente hacia adelante; no existen conexiones hacia atrás. En la figura 3.2, la primera capa recibe los datos de entrada (en este caso se tienen solamente dos entradas x_1 y x_2 , por lo tanto se tienen dos neuronas; una por cada entrada). Después se tienen tres capas ocultas cada una con tres neuronas. En la capa de salida se tiene solamente una neurona para \hat{y} que representa el resultado de la red. La cantidad de capas ocultas y la cantidad de neuronas en ellas son determinadas mediante experimentación, mientras que la cantidad de neuronas en las capas de entrada y salida es específica del problema que se intenta resolver.

En la parte de arriba de la figura 3.2 se presenta en detalle qué es lo que sucede en cada neurona de la red. Una neurona i de la capa actual se conecta con una neurona j de la capa siguiente y esta conexión tiene un peso asociado w_{ij} . Los valores de las neuronas en la capa de entrada son simplemente los datos de entrada y en general, los valores de las neuronas en las capas siguientes se calculan de la siguiente manera:

$$\phi \left(\sum_{i=1}^d w_i x_i + b \right) \quad (3.5)$$

Donde d es el número de neuronas que se conectan a la neurona actual. Es decir, para calcular el valor de una neurona j , el peso w_{ij} es multiplicado por el valor de la neurona i . Lo anterior se realiza por cada neurona que se conecte a j y después, se realiza una suma de todas estas multiplicaciones. Finalmente al resultado de esta suma se le agrega el valor de sesgo de la capa actual y esto es pasado por una función de activación ϕ . Lo anterior se realiza para cada neurona de la red.

La función de activación ϕ tiene inspiración en las neuronas del cerebro y es utilizada para decidir el nivel de activación de la neurona. Entre las funciones de activación más utilizadas se encuentran la función sigmoide, tangente hiperbólica, *softmax* y *ReLU* (por sus siglas en inglés). Todas estas funciones definen de manera diferente el resultado de la neurona y dependiendo este valor final se decide cuánto contribuirá esa neurona al resultado de la red.

Un aspecto importante que no se ha explicado es el propósito de los elementos

de sesgo b_1, \dots, b_k mostrados en la figura 3.2. Estos elementos se encuentran en todas las capas excepto en la capa de salida y normalmente tienen valor de 1 o -1. Tómese como ejemplo la función lineal $f(x) = mx + b$, en esta ecuación el término b permite mover la función hacia arriba o hacia abajo. De forma parecida funciona el sesgo en las redes neuronales; estos términos permiten mejorar el ajuste de la red.

3.3.1. Optimización

El encontrar f^* propone un problema de optimización en el cual se busca minimizar la discrepancia entre las predicciones hechas por la red en entrenamiento y los valores deseados de \mathcal{D}_E .

Supóngase una función $y = f(x)$ donde tanto x como y son números reales. La derivada de la función denotada como $f'(x)$ o $\frac{dy}{dx}$ proporciona la rapidez con la que cambia el valor de $f(x)$ en el punto x . Es decir, la derivada brinda conocimiento sobre la pendiente en un punto específico. Con esta información es posible seguir la dirección contraria a la pendiente cuando se busca minimizar una función.

La técnica conocida como **descenso por gradiente** [6] es un método de optimización muy popular utilizado en redes neuronales. Esta técnica consiste en utilizar la derivada para minimizar cualquier función de costo $f(x)$.

Descenso por gradiente

Utilizando la derivada de una función se puede tener una noción de qué dirección tomar para minimizar o maximizar su resultado. En la figura 3.3 se presenta un ejemplo de cómo utilizar la derivada para encontrar la mejor solución de una función que se desea minimizar.

Algunas veces, la derivada de una función puede no ofrecer información relevante, esto es cuando $f'(x) = 0$. Estos casos son conocidos como valores críticos. Los valores o puntos críticos pueden ser: puntos máximos locales, puntos mínimos locales o puntos de silla. En la figura 3.4 se presenta un ejemplo de cada uno de estos.

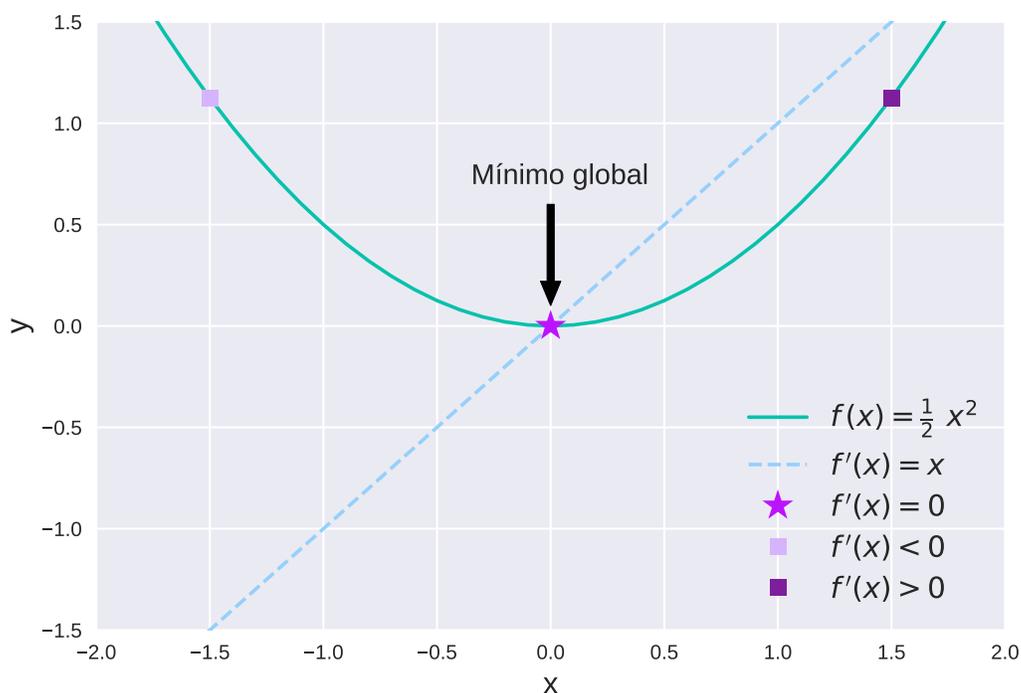


Figura 3.3: Ejemplo de descenso por gradiente en una función $f(x)$. En aquellos puntos donde $f'(x) < 0$, la variable x deberá moverse hacia la derecha para disminuir f . Asimismo, en los puntos donde $f'(x) > 0$, se debe desplazar hacia la izquierda para disminuir f , hasta alcanzar el mínimo global de la función que se encuentra donde $f'(x) = 0$.

Dependiendo de si se busca minimizar o maximizar el objetivo en cuestión, se desea encontrar un mínimo global o un máximo global. Un punto x que es mínimo global corresponde a aquél punto en donde se obtiene el menor valor posible de $f(x)$ y es posible tener uno o muchos mínimos globales para una función. Asimismo, para un punto x que es máximo global corresponde al punto en donde $f(x)$ alcanza su máximo valor posible. Cabe destacar que un punto que es mínimo o máximo local no necesariamente es mínimo o máximo global de la función.

En el contexto de aprendizaje profundo, suelen optimizarse funciones que cuentan



Figura 3.4: Casos posibles cuando $f'(x) = 0$ o tipos de puntos críticos. Cuando el punto es mayor que sus vecinos, es denominado *máximo local*, cuando el punto es menor que sus vecinos se trata de un *mínimo local* y finalmente un *punto de silla* cuando no es ninguno de los anteriores.

con muchos puntos críticos y muchos mínimos locales que no son necesariamente el mínimo global. De lo anterior surge la dificultad al tratar de optimizar este tipo de funciones y, por lo tanto, es común conformarse con soluciones para f que suelen no ser la mejor pero son suficientemente buenas.

En los problemas en donde se intentan optimizar funciones que contienen múltiples variables de entrada, se debe hacer uso de **derivadas parciales**. La derivada parcial $\frac{\partial f}{\partial x_i}$ mide cómo cambia la función f cuando solamente la variable x_i cambia. Esta generalización del concepto de derivada se presenta en el **gradiente** de una función. Denotado por ∇f , el gradiente es el vector de derivadas parciales de f . El gradiente al igual que la derivada, puede ser utilizado para elegir la dirección para moverse. En particular, el gradiente es un vector de la misma dimensión k que f y

Algoritmo 1 Descenso por gradiente estocástico

-
- 1: **Dado** un vector inicial de parámetros ω y tasa de aprendizaje α
 - 2: **para** un número de iteraciones definido **hacer**
 - 3: Tomar un conjunto de m ejemplos $\{(x_1, y_1), \dots, (x_m, y_m)\} \in \mathcal{D}_E$
 - 4: Estimar el gradiente ∇f con retropropagación
 - 5: Calcular la dirección de actualización
 - 6: Actualizar ω
-

cada elemento de él es la derivada parcial en esa dimensión:

$$\nabla f = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_k} \right] \quad (3.6)$$

Descenso por gradiente estocástico

Como ya se mencionó, los datos utilizados en aprendizaje profundo son bases de datos que contienen muchísimos ejemplos (e.g. la base de datos OpenImages [33] contiene aproximadamente 9 millones de ejemplos) por lo tanto, durante el entrenamiento del modelo suele calcularse solamente el gradiente de un conjunto pequeño de ejemplos a cada paso. Lo anterior permite realizar muchas actualizaciones y aunque no son exactas es preferible a realizar pocas actualizaciones exactas durante el entrenamiento. De este cambio, surge un nuevo algoritmo llamado **descenso por gradiente estocástico** o SGD (por sus siglas en inglés); su esencia se presenta en el Algoritmo 1 (la cuarta línea se explicará en la siguiente sección).

En la primer línea del Algoritmo 1 se menciona un parámetro muy importante. La tasa de aprendizaje α (también llamada *step size* en inglés) indica cuánto se debe mover en la dirección correcta según el valor del gradiente. El valor de este parámetro es crítico para la convergencia del algoritmo, ya que si su valor es muy grande, la optimización puede divergir y si su valor es muy pequeño, el entrenamiento tardará mucho tiempo. Una estrategia para elegir su valor es hacerlo proporcional al valor del gradiente, es decir, si la función crece mucho descender mucho y si crece poco

descender poco. Otra estrategia es encontrar manualmente el valor más bajo de α que ocasiona que la optimización diverja y establecer la tasa de aprendizaje inicial a uno ligeramente menor a este.

Cada problema es diferente y por esto no existe un valor óptimo de la tasa de aprendizaje. Aunque muchas veces encontrar su valor ideal puede involucrar una etapa de experimentación bastante extensa, existen optimizadores como *Adam* [29] en donde se intenta aprender el valor ideal de la tasa de aprendizaje durante el entrenamiento.

3.3.2. Retropropagación

En la sección anterior se explicó cómo es que se puede utilizar el gradiente para minimizar una función de costo. Sin embargo, no se explicó cómo calcular eficazmente el gradiente con respecto a sus variables de entrada en una red neuronal. En esta sección se presenta este proceso, lleva por nombre retropropagación y consiste en una aplicación recursiva de la **regla de la cadena** en cálculo.

Como ejemplo del uso de la regla de la cadena para calcular el gradiente, supóngase una ecuación sencilla del tipo $y = (11x + 10)^3$ de la cual se desea calcular $\frac{\partial y}{\partial x}$. Si se utilizan variables intermedias, la ecuación anterior puede ser reescrita como sigue:

$$a = 11x \quad b = a + 10 \quad y = b^3 \quad (3.7)$$

De estas funciones intermedias se pueden calcular sus derivadas parciales fácilmente:

$$\frac{\partial a}{\partial x} = 11 \quad \frac{\partial b}{\partial a} = 1 \quad \frac{\partial y}{\partial b} = 3b^2 \quad (3.8)$$

A partir de estas derivadas, es posible obtener $\frac{\partial y}{\partial x}$ utilizando la regla de la cadena, la cual establece que:

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial b} \frac{\partial b}{\partial a} \frac{\partial a}{\partial x} \quad (3.9)$$

Por lo tanto se tiene que:

$$\frac{\partial y}{\partial x} = 33(11x + 10)^2 \quad (3.10)$$

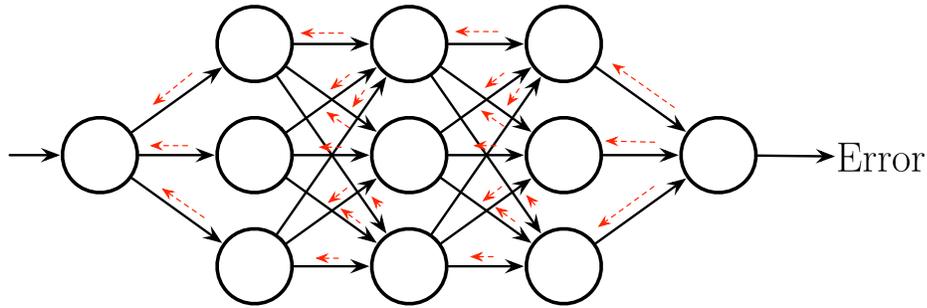


Figura 3.5: Intuición del proceso de retropropagación de errores en una red neuronal. Una vez calculado el error en la capa de salida, este se retropropaga a través de toda la red (flechas rojas) para realizar las actualizaciones apropiadas en los parámetros de la red.

Entonces, analizando el ejemplo anterior, se puede decir que multiplicando las derivadas parciales de las funciones intermedias se puede obtener la derivada de la salida de la función con respecto a la entrada.

Una **matriz Jacobiana** $\frac{\partial x_i}{\partial x_{i-1}}$ es aquella matriz conformada por las derivadas parciales de una función. Por lo tanto, con la regla de la cadena, se busca obtener el producto de todos los Jacobianos: $\frac{\partial x_k}{\partial x} = \prod_{i=1}^k \frac{\partial x_i}{\partial x_{i-1}}$ donde k es el número de funciones.

Teniendo la función de pérdida a optimizar representada mediante una red neuronal f con parámetros θ , que recibe como entradas un conjunto de ejemplos $\mathcal{D}_E = \{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(N)}, y^{(N)})\}$, se busca encontrar el gradiente $\nabla_{\theta} f$ y utilizarlo para encontrar los parámetros que optimicen el resultado de la red.

En la figura 3.5 se presenta una ilustración del proceso de retropropagación en una red neuronal. Como se mencionó, este proceso comienza en la capa de salida. Una vez calculado el error mediante una función $L(f(x), y)$, se calcula el gradiente del error con respecto a los pesos w_{ij}^k y elementos de sesgo b_i^k . De acuerdo a la dimensión de la tasa de aprendizaje α , cada iteración del algoritmo de retropropagación actualiza los pesos y elementos de sesgo (ambos representados por θ):

$$\theta^{t+1} = \theta^t - \alpha \frac{\partial L}{\partial \theta} \quad (3.11)$$

Donde θ^t denota los parámetros de la red neuronal en la iteración t del proceso de retropropagación. En concreto, el cálculo del gradiente del error con respecto a los pesos w_{ij}^k de la red se calcula mediante:

$$\frac{\partial L}{\partial w_{ij}^k} = \frac{\partial L}{\partial a_j^k} \frac{\partial a_j^k}{\partial w_{ij}^k} \quad (3.12)$$

Donde a_j^k es el resultado de la neurona j en la capa k sin haber sido pasado por la función de activación σ . De manera similar se realiza lo anterior para calcular el gradiente de los términos de sesgo de la red.

Lo anterior se debe realizar, como la figura 3.5 lo sugiere, hasta llegar a la primer capa oculta de la red. La derivada parcial en cada neurona permite determinar cuánto es que sus parámetros influyen en el error total de la red y actualizar estos parámetros en la manera correcta para minimizar la pérdida.

Como se mencionó anteriormente, en las redes neuronales esta operación se realiza de adelante hacia atrás. Las derivadas parciales de una capa son utilizadas para calcular las derivadas parciales de la capa anterior, por lo tanto es más eficaz computacionalmente realizar esta operación de la capa de salida hacia la capa de entrada.

3.4. Redes neuronales convolucionales

Al igual que las redes neuronales de propagación hacia delante mencionadas anteriormente, las redes neuronales convolucionales [36] (CNN por sus siglas en inglés) también tienen inspiración en la biología. Intentan simular el trabajo que ciertas células de la corteza visual realizan para detectar estímulos visuales como bordes, posición o tamaño de los objetos que se están viendo.

Las CNN son capaces de tomar en cuenta relaciones espaciales y locales y por esto, han sido utilizadas con éxito principalmente en tareas relacionadas a la clasificación de imágenes. En este trabajo, estas redes fueron utilizadas para el procesamiento de las imágenes en las secuencias. En la figura 3.6 se presenta un ejemplo de la arquitectura de una red convolucional.

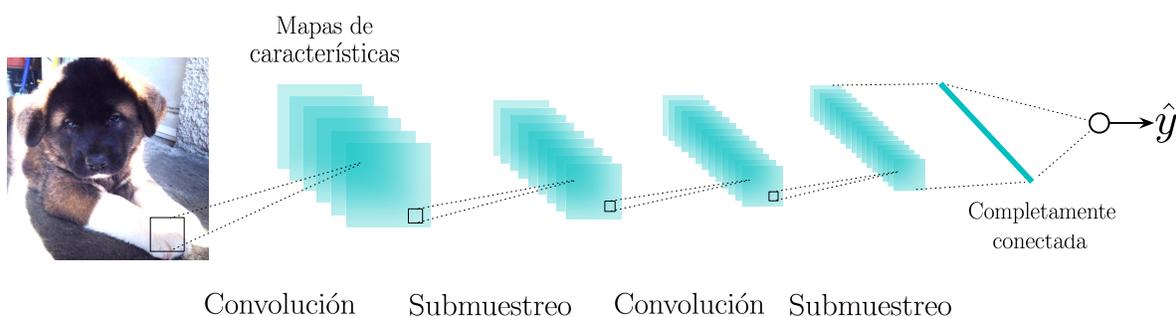


Figura 3.6: Ejemplo de una red convolucional sencilla que consiste en capas convolucionales, de submuestreo y una capa completamente conectada. Esta última capa corresponde a una capa de una red neuronal de propagación hacia delante como las explicadas en la sección anterior.

La diferencia principal de las redes convolucionales con respecto a las redes neuronales de propagación hacia delante, radica en que las capas de las redes convolucionales presentan operaciones diferentes. El nombre de las redes neuronales convolucionales tiene origen en el uso de una operación llamada **convolución**. Una convolución, en el contexto de aprendizaje profundo y redes neuronales, se entiende a grandes rasgos, como la acción de deslizar un *filtro* sobre una imagen. Este tipo de redes neuronales, al tratar con imágenes, reciben como entrada un arreglo multidimensional (llamado también **tensor**) de tamaño $\mathbf{I} = h \times w \times c$, donde h es la altura de la imagen en píxeles, w el ancho de la imagen en píxeles y c el número de canales de la imagen (usualmente 3: rojo, azul y verde). Por ejemplo, las imágenes de CIFAR-10 [34] tienen un tamaño de $32 \times 32 \times 3$, esto es 32 píxeles de alto, 32 píxeles de ancho y 3 canales pues son imágenes a color.

3.4.1. Capas convolucionales

El núcleo de una CNN son las **capas convolucionales**. Estas capas están compuestas por varios *filtros*, cada uno de los cuales es un tensor con valores que se aprenden durante el entrenamiento. El tamaño de los filtros es más pequeño que el tamaño de las imágenes, por ejemplo, un tamaño común para los filtros es de $3 \times 3 \times 3$, 3 píxeles

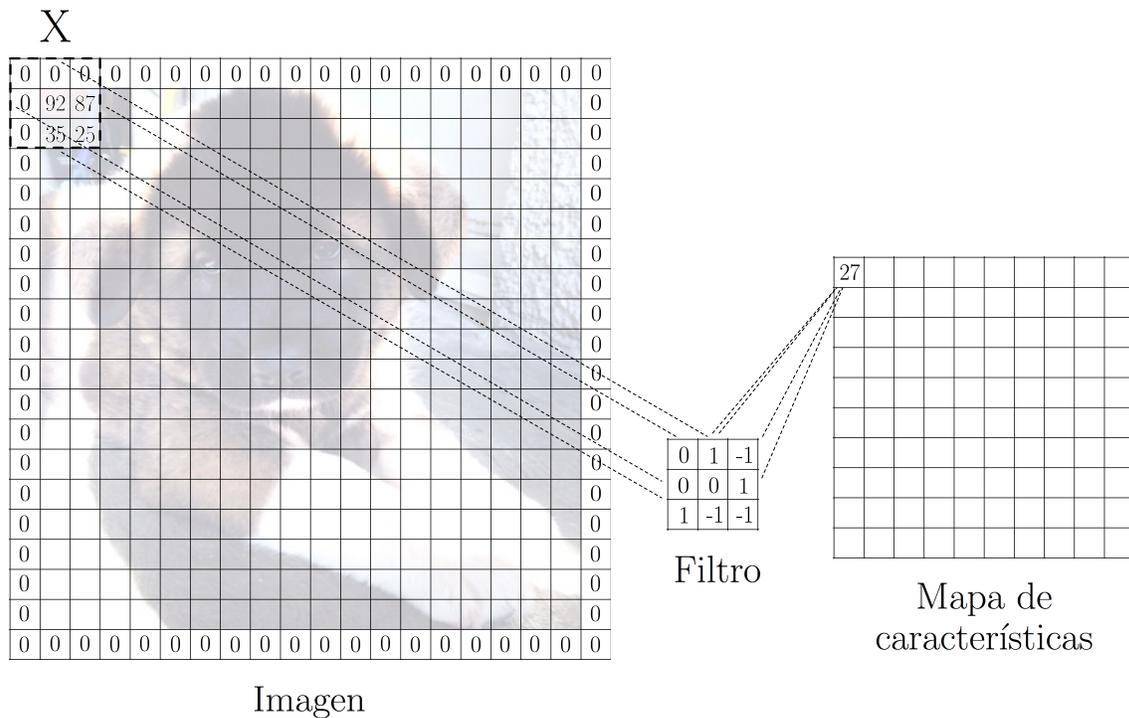


Figura 3.7: Un filtro es *convolucionado* sobre la imagen acolchada, es decir, se realiza un producto punto entre los valores del filtro con los de una porción X de la imagen. Nótese que aquí, la dimensión de profundidad no está representada por simplicidad y que en realidad para un valor del mapa de características, se toma en cuenta el producto de X y el filtro en cada una de las dimensiones de profundidad y que el resultado final (en verde) es la suma de estos tres valores (cuando son imágenes de 3 canales).

de altura, 3 píxeles de ancho y 3 canales de profundidad. Cada filtro es deslizado (o convolucionado) sobre la altura y ancho de la imagen y se realiza un producto punto entre los valores en el filtro y los valores en la pequeña ventana de la imagen.

Para deslizar el filtro se debe establecer el tamaño del *brinco*, es decir, cuántos píxeles se debe deslizar el filtro sobre la imagen. Cada vez que el filtro es deslizado sobre toda la imagen en sus tres dimensiones de profundidad, se obtiene un mapa de características que corresponde a la activación del filtro en el pedazo de la imagen en

una determinada posición.

Es una práctica común *acolchar* las imágenes con bordes de tamaño p (usualmente $p = 1$) con valor cero. Por ejemplo, si se tiene una imagen $I = n \times n$ donde $n = 6$ y un filtro $f = m \times m$ donde $m = 3$, al convolucionar el filtro sobre la imagen se obtendrá una salida de tamaño $n - m + 1 \times n - m + 1$, en este caso 4×4 . Al realizar esta operación múltiples veces, el tamaño de la representación de la imagen se verá reducida, lo cual implica una pérdida de información considerable. Es así que, por ejemplo, el hecho de acolchar la imagen con un borde $p = 1$, ocasiona que ahora la imagen sea de $I = 8 \times 8$ y por lo tanto la salida sea de tamaño $n + 2p - m + 1 \times n + 2p - m + 1$, es decir 6×6 . Lo anterior ocasiona que no se pierda información sobre los bordes de la imagen ni se reduzca la representación de la imagen. En la figura 3.7 se presenta un ejemplo de una convolución sobre una imagen acolchada con ceros.

Dado que cada capa convolucional contiene un conjunto de filtros y cada uno de estos genera un mapa de activación, se obtienen varios mapas de activación también y finalmente, estos mapas son apilados para generar un volumen final. El objetivo es que durante el entrenamiento se aprendan los mejores valores de los filtros y que con ellos, la red pueda detectar características relevantes como por ejemplo los bordes o esquinas en las imágenes.

3.4.2. Capas de submuestreo

Además de las capas convoluciones, las CNN cuentan con **capas de submuestreo** (*pooling* en inglés) que ayudan a reducir la dimensionalidad de la representación.

Las capas de submuestreo operan sobre los mapas de características y los reducen en las dimensiones de alto y ancho. Existen diferentes tipos de capas de submuestreo una de ellas es la capa *max pooling*. La función de esta capa es regresar el valor mayor de un conjunto de valores, usualmente dentro de una vecindad cuadrangular. Un tamaño común de vecindad es de 4 valores (vecindades de 2×2).

Entre otras de las operaciones que se pueden realizar están el promedio de la

vecindad (capas *avg pooling*) o la norma ℓ^2 (capas *l2-norm pooling*). La reducción de dimensión implica varias ventajas, una de ellas es que tanto el número de parámetros y el cómputo de la red disminuyen.

3.4.3. Capas completamente conectadas

La red mostrada en la figura 3.6 presenta una última **capa completamente conectada**, donde la función de esta capa es la de determinar qué valores se relacionan más con una clase determinada. Por ejemplo, si se desean clasificar imágenes de gatos y perros, esta capa tendría un tamaño de 2 representando las dos posibles clases. Para aquellas imágenes en la categoría perro, esta capa tendría valores altos para los mapas de características que representen características de perros. Los detalles de estas capas se encuentran en la sección 3.3.

3.4.4. Capas adicionales

Las capas descritas anteriormente son las capas principales utilizadas para la construcción de una red convolucional. Estas tres pueden ser utilizadas en diferentes combinaciones y pueden ser apiladas para crear redes convolucionales muy robustas las cuales suelen obtener mejores resultados. Entre otras capas que se pueden utilizar en las redes CNN están las formadas por funciones ReLU o rectificadoras. Estas funciones son usualmente utilizadas después de cada capa convolucional y su objetivo es integrar no linealidades en el modelo. Entre otras funciones rectificadoras se encuentran múltiples variantes como la *Softplus*, *Leaky ReLU* y *Noisy ReLU*, entre otras.

3.5. Redes neuronales recurrentes

Las redes neuronales recurrentes [47] (RNN por sus siglas en inglés) inicialmente presentadas en [11], presentaron una solución a múltiples tareas involucradas con el procesamiento del lenguaje natural gracias a su habilidad en el manejo de secuencias.

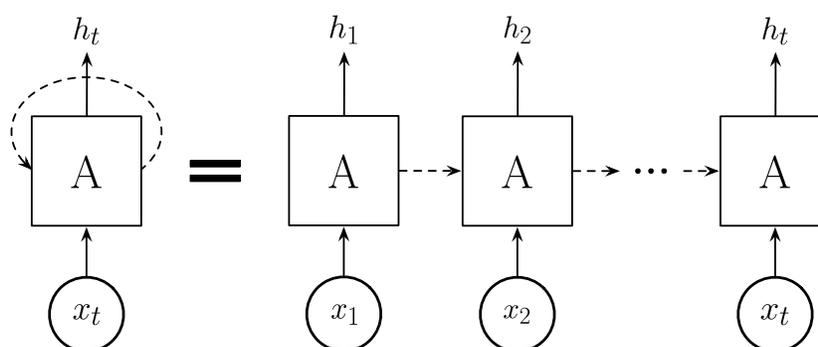


Figura 3.8: Arquitectura general de una red recurrente.

Una característica importante de las redes recurrentes es su habilidad para lidiar con secuencias de longitud variable, propiedad sustancial del lenguaje natural.

Este tipo de redes son llamadas *recurrentes* debido a que realizan la misma operación para cada elemento perteneciente a la secuencia. Una RNN modela una recurrencia de la siguiente forma:

$$h_t = f_\theta(h_{t-1}, x_t) \quad (3.13)$$

donde h_t y x_t indican la salida y entrada de la red en el instante t respectivamente, f_θ representa una función con parámetros θ que permanecen constantes durante cada instante t . Es decir entonces, que la salida generada en el momento actual, es dependiente de lo que se generó en el momento anterior y de la entrada actual.

En la figura 3.8 puede se apreciar una representación visual de lo que quiere decir la ecuación 3.13. Esta consiste de una entrada x_t suministrada a la red **A** (o función f_θ) que genera una salida h_t en el instante de tiempo t . La red contiene un ciclo que le permite utilizar información del momento anterior $t - 1$, en el momento actual t . Esta red puede verse también como una versión desenrollada, la cual consiste de varias copias de la misma red conectadas sucesivamente en donde cada copia alimenta a la siguiente.

En el contexto de este trabajo, estas redes fueron utilizadas para el manejo del texto. Entonces dado que la red espera una secuencia, por ejemplo una oración de 7

palabras, (la red en realidad recibe una representación vectorial de cada palabra) la versión desenrollada de la red tendría 7 copias, una por cada palabra en la oración.

Un ejemplo bastante conocido de aplicación de este tipo de redes es la traducción automática la cual se realiza mediante un modelo secuencial conformado por redes recurrentes [62]. Cuando se desea traducir una frase de un idioma a otro, esta traducción no puede realizarse simplemente traduciendo palabra por palabra; se debe analizar el contexto de toda la frase para realizar una traducción apropiada. Las conexiones recurrentes en la red facilitan esto y permiten traducir la palabra actual, tomando en cuenta las palabras previas a esta.

Aunque este tipo de redes recurrentes son buenas trabajando con datos secuenciales, una red recurrente simple pierde información en cuanto se le suministran secuencias muy largas y esto afecta su rendimiento [4]. Razón por la cual se crearon otro tipo de redes recurrentes capaces de trabajar con secuencias largas, tales como las LSTM (*Long Short-Term Memory*) [21] o las GRU (*Gated Recurrent Unit*) [8]. Este tipo de redes recurrentes presentan la misma estructura de las redes recurrentes comunes pero contienen compuertas que les ayudan a manejar mejor la información.

Debido a su buen rendimiento, en este trabajo se utilizaron redes recurrentes LSTM. A continuación se presenta una explicación de su funcionamiento y arquitectura.

3.5.1. *Long Short Term Memory*

Como se mencionó anteriormente, las LSTM son un tipo de red recurrente que es capaz de aprender dependencias en secuencias largas.

La diferencia de una LSTM a una red recurrente común yace en la arquitectura de la función f_{θ} o bloque **A** presente en la figura 3.8. En una red recurrente común, esta función representa simplemente una capa con la función tangente hiperbólica. En cambio, en una LSTM se tienen cuatro capas acopladas de una manera particular.

El núcleo del bloque **A** es el estado de la celda, denotado de aquí en adelante por

C_t . El resultado de este estado es influenciado por tres *compuertas* las cuales regulan su contenido durante el cómputo. Cada una de estas compuertas tiene sus matrices de parámetros (i.e. pesos y sesgos) que deben ser aprendidas durante el entrenamiento. A continuación se explican cada una de estas compuertas.

Compuerta de olvido

El propósito de esta compuerta es determinar qué información conservar. En esta primer compuerta se calcula lo siguiente:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (3.14)$$

donde h_{t-1} representa el estado anterior, x_t la entrada del instante actual, W_f y b_f representan la matriz de pesos y el sesgo de f , respectivamente. Y finalmente σ representa la función de activación sigmoide. Nótese que en todos los instantes t , tanto W_f como b_f permanecen invariables. Esta compuerta decide qué información se debe conservar analizando la información generada anteriormente h_{t-1} y la información actual x_t . La función sigmoide se encarga de regresar un valor $r \in [0, 1]$ el cual representará qué tanta información debe ser conservada (1 indica conservar toda la información y 0 indica olvidar todo).

Compuerta de entrada

Después, se debe decidir qué nueva información se añadirá a la celda. Esta decisión está compuesta por dos fases. En la primer fase participa la compuerta de entrada, denotada por la ecuación 3.15. Su trabajo es decidir qué información se deberá actualizar:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (3.15)$$

Nótese que la ecuación de esta compuerta es muy parecida a la ecuación de la compuerta de olvido 3.14, sin embargo, en esta se utilizan la matriz de pesos y el ses-

go de la compuerta correspondiente. En la segunda fase, se crea un posible candidato de nuevos valores \tilde{C} que podría ser agregado al estado de la celda:

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (3.16)$$

Asimismo, se utilizan la matriz de pesos y sesgo correspondiente. Para actualizar el estado de la celda, se toman en cuenta las tres ecuaciones anteriores:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (3.17)$$

Aquí se combinan el valor anterior de la celda y f_t . Esto quiere decir que una vez olvidada la información innecesaria, se agregan los valores que fueron considerados relevantes mediante las ecuaciones 3.15 y 3.16. La ecuación anterior obtiene el nuevo valor de la celda.

Compuerta de salida

El trabajo de esta compuerta es básicamente dejar salir una versión resumida de lo obtenido por la ecuación 3.17. Se debe decidir qué partes del estado de la celda formarán parte de la salida, así que se utiliza una función sigmoide para decidirlo:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (3.18)$$

Finalmente, el nuevo estado C_t es pasado por una función tangencial hiperbólica. Esto simplemente para hacer que sus valores se encuentren en el rango $[-1, 1]$. Lo anterior es combinado con las partes elegidas por 3.18, constituyendo así la salida de la celda:

$$h_t = o_t * \tanh(C_t) \quad (3.19)$$

Aparte de las LSTM existen otros tipos de redes recurrentes que han sido creadas en la búsqueda para obtener un mejor rendimiento. Entre las más destacadas se pueden encontrar las LSTM *peephole* [14], la GRU (*Gated Recurrent Unit*) [8] y LSTM

convolucional [49]. En algunos trabajos se han realizado comparaciones de muchos tipos de redes recurrentes [18, 26]. En [26] se encontró que en tareas específicas otras redes pueden tener un mejor rendimiento que las LSTM.

3.5.2. Entropía cruzada

El entrenamiento de una red neuronal involucra tener una métrica que sea capaz de evaluar el desempeño del modelo adecuadamente, tal como se mencionó en la sección 3.1. Una métrica de pérdida común utilizada en redes neuronales recurrentes y modelos que lidian con tareas de generación de texto es el *error por entropía cruzada*. El error por entropía cruzada está denotado por:

$$E(\hat{y}, y) = -\frac{1}{N} \sum_{n \in N} y_n \log \hat{y}_n \quad (3.20)$$

donde N representa el número de ejemplos en los datos de entrenamiento \mathcal{D}_E , y la respuesta esperada y \hat{y} la predicción hecha por la red. Básicamente, es el error acumulado de todos los datos de entrenamiento; mientras más alejada esté la palabra predicha de la palabra esperada, el error será mayor.

3.5.3. Retropropagación a través del tiempo

Las redes recurrentes utilizan los mismos pasos de entrenamiento que las redes neuronales simples, pero dado que las RNN presentan una arquitectura muy peculiar, para su entrenamiento se utiliza una modificación del método de retropropagación presentado en la sección 3.3.2.

El método de retropropagación a través del tiempo se realiza sobre la red recurrente desenrollando la red (como se muestra en la figura 3.8) en k copias de la red, donde k es el tamaño de la secuencia que la red está recibiendo.

Dado que en cada instante de tiempo t la red recurrente genera una salida \hat{y}_t , cada una de estas salidas son comparadas con su correspondiente salida esperada y_t y el

error final es la suma de cada uno de estos pequeños errores. Por lo tanto, el error queda modelado como:

$$E_t(\hat{y}_t, y_t) = -y_t \log \hat{y}_t \quad (3.21)$$

$$E(\hat{y}_t, y_t) = \sum_t E_t(\hat{y}_t, y_t) \quad (3.22)$$

$$= - \sum_t y_t \log \hat{y}_t \quad (3.23)$$

Nótese que el error anterior es el error de cada ejemplo en los datos de entrenamiento, es decir, una secuencia completa es considerada un ejemplo. Recordando, una vez calculado el error final en el entrenamiento de la red, se utiliza el gradiente para saber qué modificaciones realizar a los parámetros de la red que minimicen el error. El cálculo del gradiente en una red recurrente presenta el mismo cambio que en el cálculo del error de la ecuación 3.21. Para cada ejemplo, se suman los gradientes de cada instante de tiempo t :

$$\frac{\partial E}{\partial \theta} = \sum_t \frac{\partial E_t}{\partial \theta} \quad (3.24)$$

3.6. Modelos secuencia a secuencia

En la búsqueda de modelos de procesamiento del lenguaje, múltiples arquitecturas han sido concebidas. Una de ellas, en la cual se basa el método presentado en este trabajo, es el modelo secuencia a secuencia o *Seq2Seq* [7], [52]. La ventaja de esta arquitectura es que permite mapear una secuencia de entrada a una secuencia de salida, las cuales no son necesariamente del mismo tamaño. Esta arquitectura ha sido utilizada con éxito en tareas de traducción automática [63], generación de resúmenes automáticos, entre otros.

Un modelo secuencia a secuencia consiste en dos redes recurrentes, un codificador y un decodificador. En la figura 3.9 se presenta su arquitectura.

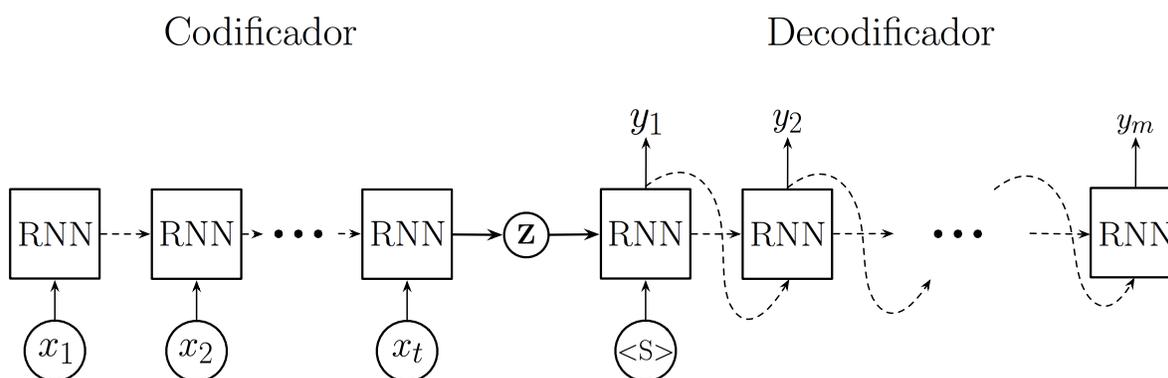


Figura 3.9: Arquitectura general de un modelo secuencia a secuencia. El codificador procesa una secuencia de entrada $\{x_1, x_2, \dots, x_t\}$ y captura en su último estado oculto (aquí representado por z) el contexto de la secuencia de entrada. El decodificador utiliza la información en z para generar una secuencia de salida $\{y_1, y_2, \dots, y_m\}$. El *token* de inicio denotado aquí por $\langle S \rangle$ es la primer entrada del decodificador y es utilizado para empezar la generación de la secuencia.

La noción de un *Seq2seq* es la siguiente: la primer RNN usualmente llamada codificador, procesa una secuencia de entrada de tamaño t . Al terminar, el codificador genera un contexto (el último estado oculto de la RNN) el cual es un tensor de tamaño fijo que representa el resumen o la esencia de la secuencia de entrada. Finalmente, la segunda RNN llamada decodificador, es condicionada a este contexto para generar una secuencia de salida de tamaño m . Tanto el codificador como el decodificador son entrenados para maximizar la probabilidad $p(y_1, y_2, \dots, y_m | x_1, x_2, \dots, x_t) \forall (x, y) \in \mathcal{D}_E$.

4

Generando relatos

En este capítulo se describe en detalle el método propuesto para generar relatos a partir de secuencias de imágenes. El método que se propone utiliza como base el sistema *Show and Tell* presentado en [60]. Aunque este sistema no está diseñado para generar relatos de una secuencia de imágenes, posee ciertos componentes que pueden ser utilizados para esta tarea. El método presentado más adelante en esta sección extiende *Show and Tell* a una arquitectura secuencia a secuencia como la presentada en la sección 3.6.

4.1. *Show and Tell*

En esta sección se presenta un panorama general de la arquitectura de *Show and Tell* que fue utilizada como base para el modelo propuesto. Dado que este modelo realiza descripciones de imágenes, no es factible utilizar directamente su arquitectura para la generación de relatos. Debido a esto, se realizaron algunas modificaciones a la arquitectura para convertirla en un modelo secuencia a secuencia. Estos cambios permitieron generar relatos a partir de una secuencia de imágenes.

En concreto, *Show and Tell* está compuesto por una red neuronal convolucional llamada *Inception V3* [53] y una red recurrente LSTM. En la figura 4.1 se presenta esta

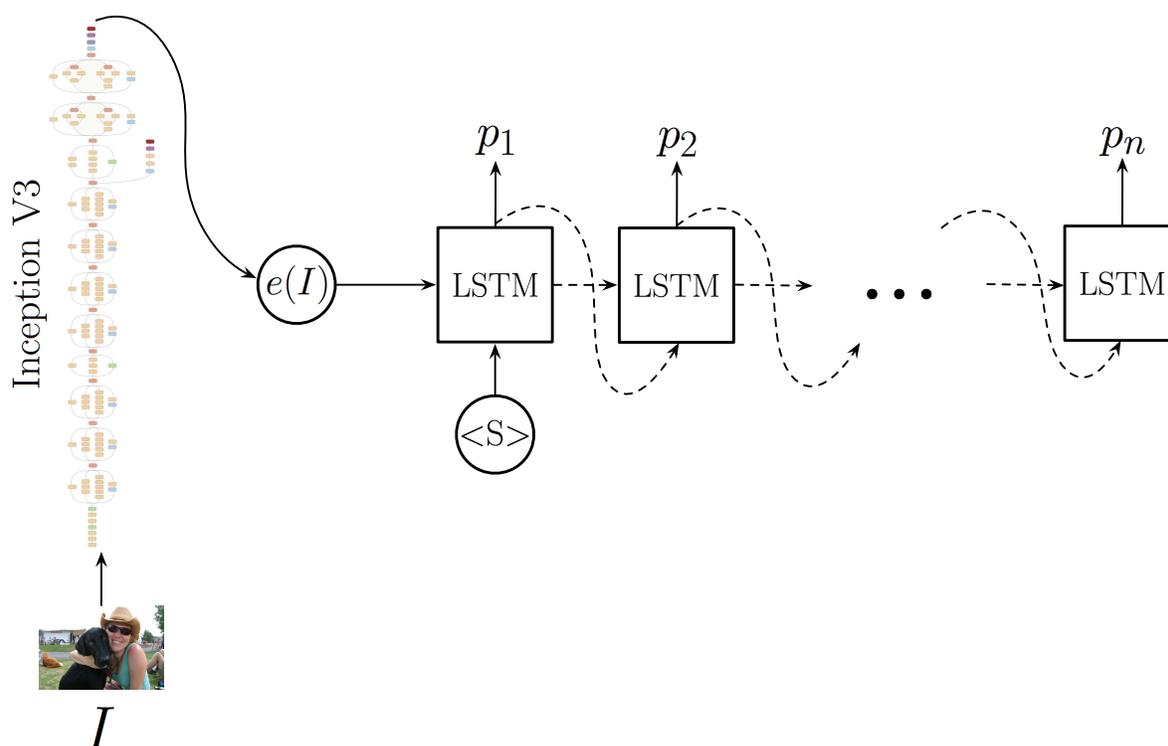


Figura 4.1: Arquitectura de *Show and Tell*. La red convolucional *Inception V3* recibe la imagen a describir y la codifica en una representación $e(I)$ que es utilizada para inicializar el primer estado de la red LSTM. La red recurrente recibe una señal de inicio denotada aquí por $\langle S \rangle$ y empieza a generar la descripción de la imagen $\{p_1, p_2, \dots, p_n\}$ una palabra en cada instante a utilizando la información en $e(I)$.

arquitectura. La red convolucional recibe la imagen a describir I y la codifica en una representación vectorial $e(I)$ que es utilizada para inicializar el primer estado ($h^{(0)}$) de la red LSTM. Solamente durante el entrenamiento, la red recibe como entradas las palabras de referencia y durante la etapa de inferencia, la red recurrente genera la descripción de la imagen $\{p_1, p_2, \dots, p_n\}$, una palabra por cada instante.

En la arquitectura de *Show and Tell* se tiene un panorama codificador-decodificador, esto quiere decir que la red convolucional codifica la imagen y la red recurrente intenta decodificarla en una secuencia de palabras. Para la generación de relatos se tiene un panorama secuencia a secuencia. En concreto, se utilizó la misma red convolucional

Inception V3 de *Show and Tell*, se utilizaron varias copias de la red LSTM presentada en *Show and Tell* para la generación de texto y finalmente se agregó otra red LSTM con la misma arquitectura que las otras para codificar la secuencia de imágenes. A continuación se detallan los componentes del método propuesto para esclarecer las modificaciones realizadas.

4.2. Componentes principales

Inicialmente, se explicará la red utilizada para obtener la representación de las imágenes, después se describe la arquitectura del método en función de sus dos componentes principales, el **codificador** y el **decodificador**. Finalmente, uniendo estos componentes se presenta el método completo y se explica su comportamiento durante las fases de entrenamiento e inferencia.

4.2.1. *Inception V3*

Para realizar relatos a partir de imágenes se requiere de una red convolucional que sea capaz de extraer información importante de las imágenes que posteriormente pueda ser utilizada en los relatos. *Inception V3* es una red convolucional que presenta una arquitectura muy compleja y que ha demostrado tener buenos resultados en tareas de clasificación de imágenes [53]. Esta misma red es utilizada en *Show and Tell* [60] para extraer la información relevante de las imágenes.

Esta red participó en la competencia *Large Scale Visual Recognition Challenge* del año 2015, en la cual el objetivo principal era el de clasificar imágenes en 1,000 posibles categorías. En comparación con otras arquitecturas convolucionales (*AlexNet*, *VGGNet*, entre otras), *Inception V3* reduce el número de parámetros a optimizar, teniendo un efecto positivo en su desempeño en general. Por tal motivo, es utilizada comúnmente en aplicaciones que manejan una gran cantidad de imágenes. La tarea de la red es entonces, mediante sus capas, extraer las características de una imagen,

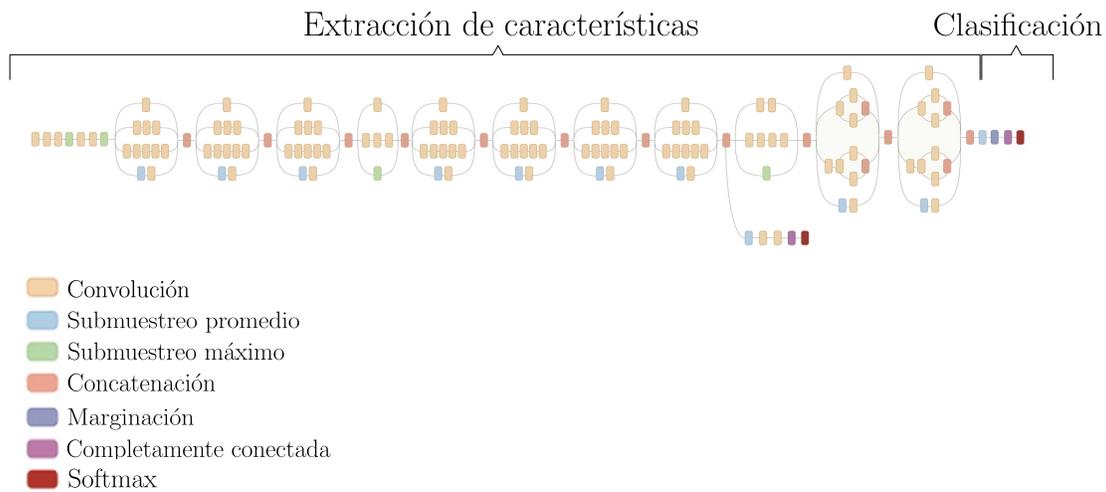


Figura 4.2: Arquitectura de *Inception V3*. La etapa de extracción de características está constituida por capas convolucionales, capas de submuestreo y capas de concatenación, estas últimas unen los resultados de las capas previas en un solo tensor. La etapa de clasificación se constituye de capas completamente conectadas y capas *Softmax*.

representarlas en un tensor con tamaño fijo de valores continuos, para después clasificarla.

Inception V3 en general consta de dos etapas principales, extracción de características y clasificación. La característica principal de esta red son los módulos *inception* de los que hace uso. En la figura 4.2 se presenta su arquitectura, mientras que las capas de la red se encuentran detalladas en el cuadro 4.1.

La entrada de la red se asume de la siguiente manera:

- Un tensor de dimensión $w \times h \times 3$ donde w representa el ancho de la imagen en píxeles, h representa el alto de la imagen en píxeles y 3 representa el número de canales de la imagen, (en este caso 3 pues todas las imágenes se encuentran en RGB). Tanto h como w en *Inception* están definidos con el valor de 299, como se muestra en el cuadro 4.1.

En la aplicación real, la red tiene como entrada un conjunto (también llamado lote) de imágenes, es decir, un conjunto de estos tensores. La función de error que

Capa	Tamaño de filtro y de paso	Tamaño de entrada
Convolución	$3 \times 3, 2$	$299 \times 299 \times 3$
Convolución	$3 \times 3, 1$	$149 \times 149 \times 32$
Convolución acolchada	$3 \times 3, 1$	$147 \times 147 \times 32$
Submuestreo	$3 \times 3, 2$	$147 \times 147 \times 64$
Convolución	$3 \times 3, 1$	$73 \times 73 \times 64$
Convolución	$3 \times 3, 2$	$71 \times 71 \times 80$
Convolución	$3 \times 3, 1$	$35 \times 35 \times 192$
$3 \times Inception$	Como en la figura 4.3	$35 \times 35 \times 288$
$5 \times Inception$	Como en la figura 4.4	$17 \times 17 \times 768$
$2 \times Inception$	Como en la figura 4.5	$8 \times 8 \times 1280$
Submuestreo	8×8	$8 \times 8 \times 2048$
Lineal	<i>logits</i>	$1 \times 1 \times 2048$
<i>Softmax</i>	clasificador	$1 \times 1 \times 1000$

Cuadro 4.1: Detalle de las capas en *Inception V3*. Reproducido de [53].

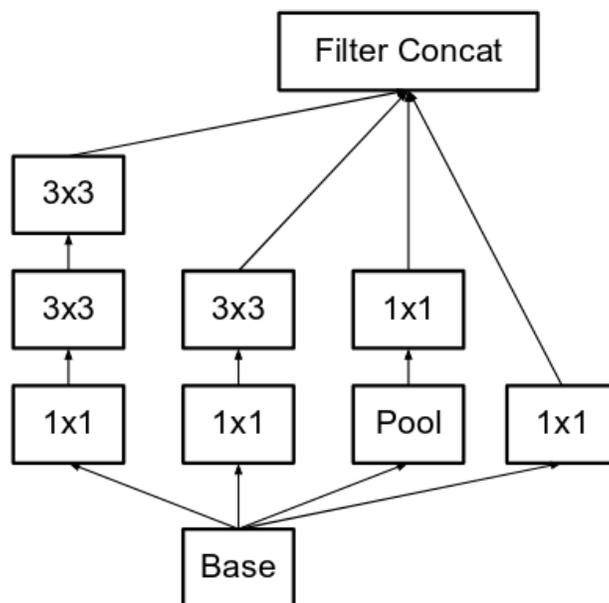


Figura 4.3: Se muestra un ejemplo de los módulos *inception* utilizados en la arquitectura de *Inception V3* en donde cada convolución de 5×5 es reemplazada por una convolución de 3×3 . Reproducido de [53].

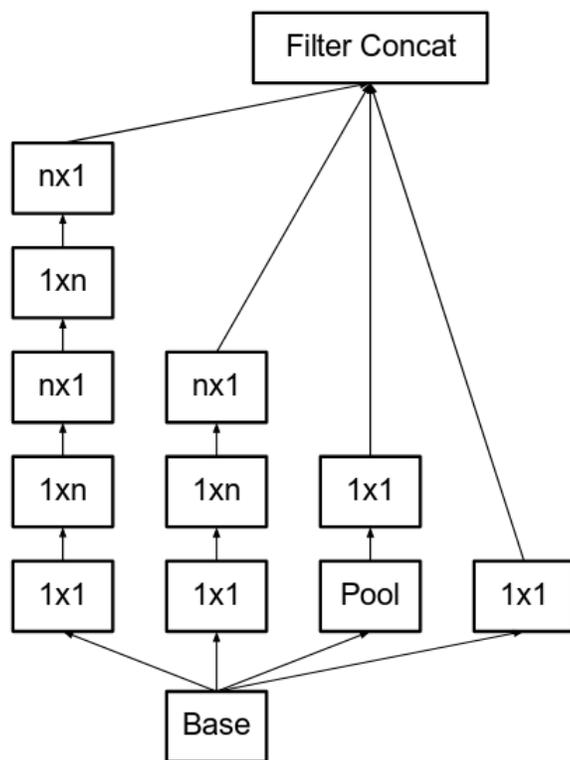


Figura 4.4: Detalle de módulo *inception* después de la factorización de una convolución de $n \times n$, donde $n = 7$. Reproducido de [53].

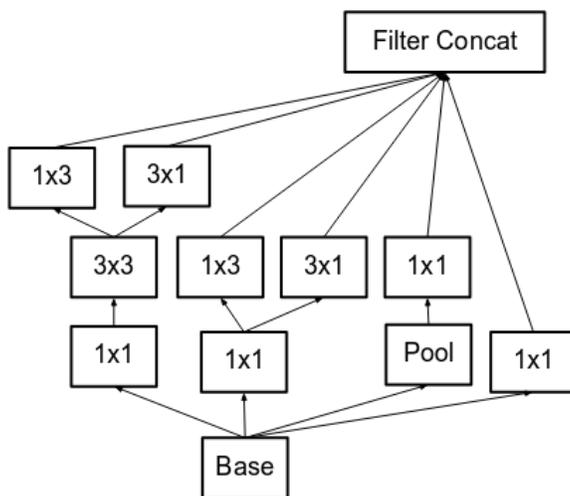


Figura 4.5: Factorización de una convolución de 8×8 . Este módulo es utilizado para promover representaciones de alta dimensionalidad. Reproducido de [53].

emplea *Inception V3* para ajustar su entrenamiento es la función de entropía cruzada definida en la ecuación 3.20.

En tareas en las que no se desea clasificar imágenes sino simplemente extraer características de las imágenes, como en la que se aborda en este trabajo, *Inception V3* puede ser utilizada simplemente utilizando la parte de extracción de características de la red para obtener información de las imágenes. De esta etapa se obtiene como salida el vector de características de la imagen con dimensión de 2048 como se muestra en el cuadro 4.1. Sin embargo, esta dimensión puede ser modificada para otras tareas si así se desea. En este proyecto se decidió utilizar una dimensión de 512 por conveniencia, la cual será discutida más adelante. Se utilizó la misma implementación de *Inception V3* presente en el código de *Show and Tell* [60].

4.2.2. Extrayendo la esencia

El primer componente del método propuesto está compuesto por una red neuronal recurrente que se encarga de resumir la secuencia de imágenes. Esta red es una de las adiciones a la arquitectura de *Show and Tell* para poder generar relatos de una secuencia de imágenes. Antes de explicarlo en detalle, se discutirá la entrada del codificador que se conforma de la siguiente manera:

- **Secuencia de imágenes** Un tensor de dimensiones $b \times a \times c$ donde b representa el número de secuencias a procesar (tamaño del lote), a es el número de imágenes en la secuencia (en este caso su valor es siempre 5 pues todas las secuencias contienen 5 imágenes) y c es el tamaño de la representación de las imágenes (definido como 512 en *Inception*).

Siguiendo la explicación de los modelos secuencia a secuencia presentada en la sección 3.6, la tarea del codificador es entonces tomar la secuencia de entrada y codificarla en un tensor de tamaño fijo que representa su esencia o resumen. En la figura 4.6 se presenta una representación del codificador del método que consiste en una red LSTM igual a las explicadas en la sección 3.5.

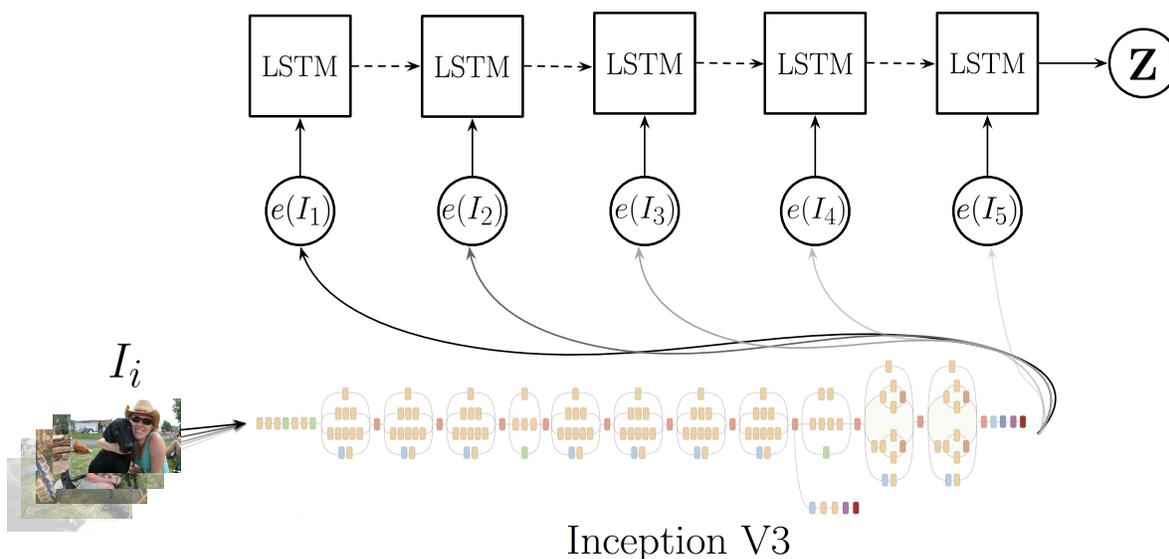


Figura 4.6: Codificador del método. La entrada del codificador es una secuencia de 5 imágenes. Cada una de ellas está representada mediante un tensor $e(I_i)$ donde $i = \{1, \dots, 5\}$ de valores continuos obtenido mediante *Inception V3* que contiene información de la imagen correspondiente. La red recurrente recibe la representación de cada imagen $e(I_i)$ en cada instante de tiempo t y cuando $t = 5$, se ha codificado la secuencia completa. Al finalizar, se obtiene el último estado del codificador \mathbf{Z} que contiene la información o esencia de la secuencia completa.

La red procesa cada imagen de la secuencia y al terminar se obtiene su contexto localizado en el último estado oculto de la LSTM denotado como \mathbf{Z} en la figura 4.6. En concreto, este estado es un tensor de dimensión $b \times e$, donde b es el tamaño del lote y e es el número de unidades de la LSTM definido como 512.

Dado que esta red recurrente solamente es utilizada para obtener la esencia de la secuencia, el único componente que es de interés es el último estado oculto de la red y por esto, las salidas son ignoradas.

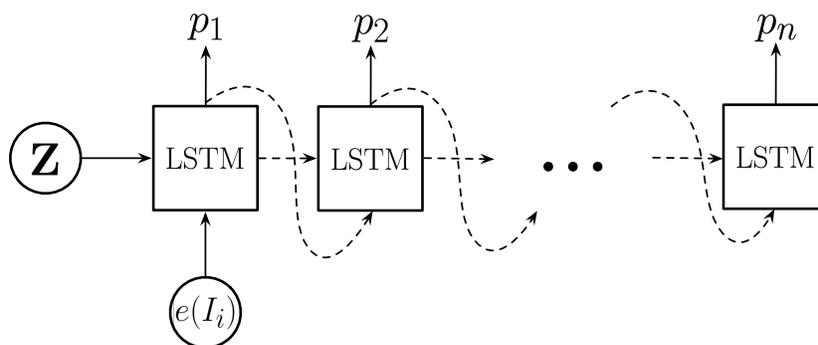


Figura 4.7: Decodificador del método. El estado del decodificador es inicializado con el último estado oculto del codificador \mathbf{Z} . Asimismo, la primer entrada del decodificador es la representación de la imagen $e(I_i)$ de la secuencia. De esta manera, el decodificador genera una secuencia de palabras $\{p_1, p_2, \dots, p_n\}$ la cual está condicionada al contenido de la secuencia completa \mathbf{Z} y a la información de la imagen $e(I_i)$. Es importante destacar que por eficiencia, al obtener las salidas del decodificador estas son pasadas por una red completamente conectada y en modo inferencia por la función *softmax* para obtener las predicciones; por simplicidad, estas redes no están representadas en la figura.

4.2.3. Generando un relato

El segundo elemento de la arquitectura propuesta está compuesto también por una red neuronal recurrente LSTM, la cual es la encargada de generar los relatos utilizando la información obtenida mediante el codificador. Esta red pretende realizar la misma función que el decodificador en *Show and Tell*; generar una secuencia de palabras a partir de la información en una representación obtenida con *Inception V3*. Al igual que en la sección anterior, primero se definirán los elementos de entrada del decodificador:

- **Imagen** - Un tensor de dimensión $b \times c$ donde b representa el número de imágenes a procesar (en este caso solamente 1) y c es el tamaño de la representación de la imagen. Este tensor es el vector de características de la imagen $e(I_i)$.

- **Z** - Último estado oculto generado por el codificador. Es un tensor de dimensión $b \times e$ donde b es el tamaño del lote y e es el número de unidades de la LSTM. Nótese que el estado oculto de una red recurrente contiene dos elementos en realidad, c y h mencionados en la sección 3.5, entonces tanto c como h son tensores de esta dimensión.

Aunque el codificador y el decodificador son redes LSTM con la misma estructura, el decodificador presenta una inicialización personalizada (a diferencia de inicializarla con ceros como comúnmente se realiza). La tarea del decodificador es entonces, generar una salida utilizando la información provista por el codificador. En la figura 4.7 se presenta una representación del decodificador del método. El último estado oculto del codificador es utilizado como el estado inicial ($h^{(0)}$) del decodificador y la primer entrada del decodificador es la representación de la imagen de la cual la red debe generar una secuencia de palabras. Esta secuencia de palabras será la parte correspondiente a la imagen que contribuye al relato completo de la secuencia.

4.3. Método propuesto

Una vez teniendo claro cada uno de los componentes, se puede presentar la arquitectura completa. La figura 4.8 presenta la unión de los elementos explicados anteriormente.

El decodificador discutido en la sección anterior, toma como entrada solamente una imagen y genera la secuencia de palabras para esa imagen, tal como se tiene en la arquitectura de *Show and Tell*. Sin embargo, lo que se busca es generar un relato para una secuencia, así que esto se propone realizarlo haciendo uso de múltiples decodificadores; un decodificador para cada imagen de la secuencia. De esta manera, cada decodificador generará la parte textual correspondiente a cada imagen y el relato de la secuencia completa será la unión de la salida de cada uno de los decodificadores.

A continuación se describe el flujo del método:

- Se proporciona como entrada una secuencia de 5 imágenes $A = \{I_1, I_2, \dots, I_5\}$ de la que se desea generar un relato.
- Cada una de las imágenes de la secuencia es transformada a un tensor de dimensión $299 \times 299 \times 3$ y es pasado por *Inception V3* para obtener su tensor de características $e(I_i)$.
- El codificador es una red LSTM recibe como entrada el conjunto de representación de las imágenes $\{e(I_1), \dots, e(I_5)\}$ y computa su último estado oculto z al terminar de procesar la secuencia de imágenes.
- El decodificador d_i , toma como primer entrada la representación de la imagen $e(I_i)$ de la secuencia y se inicializa su primer estado oculto ($h_{d_i}^{(0)}$) con el contexto de la secuencia z . A partir de esto, el decodificador genera una palabra p en cada instante de tiempo t , hasta generar la secuencia de palabras completa $\{p_1, p_2, \dots, p_{n_i}\}$ donde n_i representa la longitud de la secuencia de palabras para la imagen I_i .

4.3.1. Fase de entrenamiento

La arquitectura se entrena para predecir cada palabra del texto de cada imagen de la secuencia. Esto lo realiza tomando en cuenta la imagen en cuestión y las palabras anteriores que ha generado. Durante el entrenamiento del método se desarrollan las recurrencias LSTM (codificador y múltiples decodificadores), el codificador en 5 copias de la red (una por cada imagen) y cada decodificador en n_i copias, donde n_i es la longitud de la secuencia de palabras generadas por el decodificador d_i .

En cada decodificador, en cada instante de tiempo t se calcula la probabilidad de que la palabra w sea la palabra para el instante actual S_t , lo cual se realiza para todas las palabras en el vocabulario a partir de $p(S_t|I, S_0, \dots, S_{t-1})$. Es importante notar que la primer palabra S_0 y la última palabra S_n generadas por el modelo son *tokens* especiales que denotan el inicio y fin de oración, respectivamente. El generar el *token*

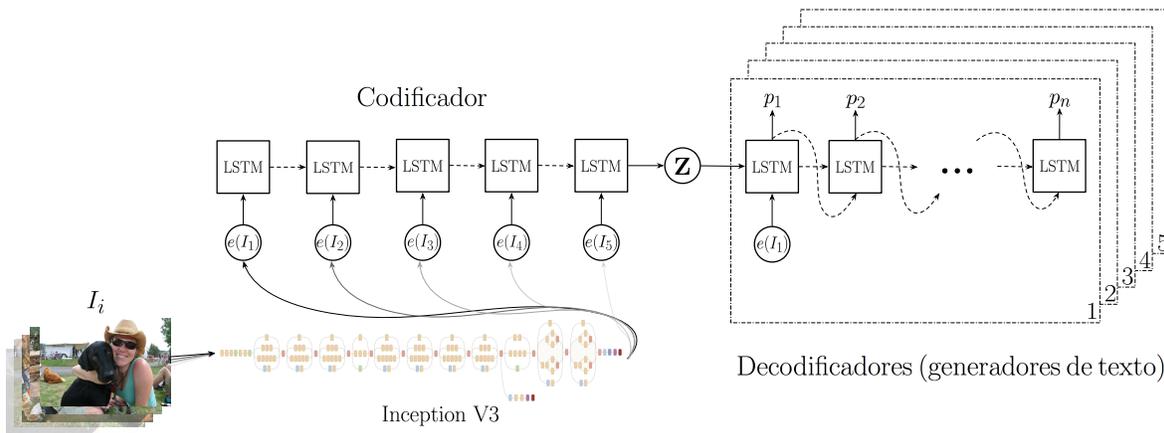


Figura 4.8: Método propuesto. Se tiene la red para representación de las imágenes *Inception V3*, una red recurrente como codificador y 5 redes recurrentes como decodificadores, uno para cada imagen. Cada decodificador d_i toma el último estado del codificador para inicializar su estado inicial y recibe como primer entrada la representación $e(I_i)$ de la imagen. El decodificador d_i genera la parte textual que corresponde a la imagen I_i . El relato final es la unión de la salida de los 5 decodificadores.

de fin de oración funge como señal de que una secuencia de palabras completa ha sido generada y se debe terminar la recurrencia del decodificador.

Sea I la imagen en cuestión y $S = (S_0, \dots, S_n)$ una secuencia correcta de palabras que conforma el relato de la imagen, el flujo en cada decodificador d_i puede definirse formalmente de la siguiente manera:

$$h^{(0)} = h_c^{(t)} \quad (4.1)$$

$$x_{-1} = e(I) \quad (4.2)$$

$$x_t = W(S_t), t \in \{0, \dots, n-1\} \quad (4.3)$$

$$p_{t+1} = d_i(x_t), t \in \{0, \dots, n-1\} \quad (4.4)$$

donde $h^{(0)}$ y x_{-1} indican el primer estado y la primer entrada del decodificador respectivamente y x_t son las entradas siguientes. Dado que tanto la imagen I como las palabras S_t son utilizadas como entradas para el decodificador, deben ser mapeadas

al mismo espacio vectorial, por lo tanto cada palabra S_t es representada mediante $W(S_t)$ el cual es un tensor de la misma dimensión que $e(I_i)$. Finalmente, p_{t+1} es un tensor de probabilidades logarítmicas por cada palabra en el diccionario $d_i(x_t)$ en el tiempo $t + 1$.

La representación de cada palabra del vocabulario que se utiliza para generar la secuencia de palabras es un tensor de la misma dimensión que la representación de las imágenes, razón por la que ambas dimensiones fueron definidas como 512. La representación de las palabras fue obtenida mediante *Word2Vec* [42] debido a su buen rendimiento en *Show and Tell* [60]. Se utilizó la misma implementación de *Word2Vec* presente en el código de *Show and Tell* [60].

La optimización del entrenamiento se lleva a cabo basándose en el negativo de la probabilidad logarítmica con respecto a los parámetros de la arquitectura. Esta función de error está definida como:

$$E(I, S) = - \sum_{t=1}^n \log p_t(S_t) \quad (4.5)$$

Cuando el método ha terminado su entrenamiento (lo cual sucede cuando ha entrenado por un número de épocas predefinidas), se encuentra listo para generar relatos a partir de secuencias.

4.3.2. Fase de inferencia

En la fase de inferencia, una vez que se han aprendido los parámetros óptimos para realizar generación de relatos, dada una secuencia de imágenes como entrada se genera un relato. Hasta este momento no se ha discutido la forma en la que la arquitectura genera las frases como tal, esto lo realiza haciendo uso del algoritmo *Beam Search*.

Beam Search

Beam Search es un algoritmo voraz que explora un grafo y expande aquél nodo que considere más prometedor. El algoritmo construye el árbol de búsqueda mediante la técnica de búsqueda en amplitud. En cada nivel del árbol, se generan todos los posibles sucesores de los estados de ese nivel y son ordenados de acuerdo a su probabilidad. Este algoritmo mantiene solamente las k (también llamado en inglés *beam width*) mejores oraciones que han sido generadas hasta el nivel actual como posibles candidatas para generar la siguiente oración. Asimismo, a cada instante, solamente se mantienen las k mejor nuevas oraciones de todas las generadas. *Beam Search* regresa la primer solución encontrada.

El valor de k puede ser conservado fijo o variable. Cuando k es muy grande ningún nodo es descartado y por lo tanto, *Beam Search* es igual a una simple búsqueda en amplitud. Dado que *Beam Search* podría eliminar un nodo exitoso (aquél que brinda la solución más probable) en el proceso de búsqueda, es considerado no óptimo. En algunos casos, el algoritmo se detiene al alcanzar una profundidad máxima definida (e.g. la longitud máxima de la oración) y evalúa todas las posibles soluciones generadas a diferentes profundidades del árbol y regresa aquella solución con la probabilidad más alta.

Como se mencionó anteriormente, el seleccionar el *token* de fin de oración funciona como una bandera para saber que una secuencia de palabras completa ha sido encontrada.

5

Experimentación y evaluación

La primera parte de este capítulo se enfoca en el conjunto de datos utilizado para el entrenamiento y evaluación del método. En la siguiente sección se detallan las métricas que fueron utilizadas para la evaluación de los relatos generados.

Finalmente, se presenta la evaluación de dichos relatos. Esta evaluación fue llevada a cabo de dos maneras: la primera se centra en la participación en el *Visual Storytelling Challenge* que tuvo lugar como parte del *Storytelling Workshop* en el marco del congreso del 2018 de la *North American Chapter of the Association for Computational Linguistics* (NAACL) y la segunda consta de la evaluación de los relatos haciendo uso de las métricas automáticas, sobre todo el conjunto de prueba de VIST.

5.1. Conjunto de datos

La base de datos que se utilizó para entrenar y evaluar la arquitectura es el conjunto de datos llamado VIST (*Visual Storytelling Dataset*) presentado en [54]. Este conjunto de datos se encuentra libre para su descarga y uso en su sitio oficial [25].

Creado por varios investigadores de las universidades Carnegie Mellon, Johns Hopkins, Virginia Tech y Rochester, así como de Microsoft Research y Facebook AI Research, VIST es presentado como el primer conjunto de datos que puede ser utili-

Imagen	Descripción	Relato
	<p><i>"A woman in sunglasses hugging a black dog."</i></p>	<p><i>"There are dog lovers and then there are dog lovers. And then there is my sister"</i></p>
	<p><i>"A man is sitting in a park with his black dog."</i></p>	<p><i>"and her husband. They have two kids, both are black labs."</i></p>
	<p><i>"A woman adjusting the gear that her black dog is wearing."</i></p>	<p><i>"And both know just how to get what they want."</i></p>
	<p><i>"The dog laying on the ground is black and has a red collar."</i></p>	<p><i>"They are so playful. They may be big but they are just a couple of toddlers rolling around in the grass."</i></p>
	<p><i>"A black dog that has been put a blue white and red ribbon."</i></p>	<p><i>"And they are in no way spoiled... Um yeah...ok."</i></p>

Cuadro 5.1: Ejemplo de imágenes y texto en VIST. Se tienen descripciones para cada imagen y relatos para secuencias de imágenes que consisten en pequeñas historias de 5 frases (en su mayoría), una para cada imagen de la secuencia.

zado para generar relatos a partir de secuencias de imágenes.

5.1.1. Descripción

VIST está conformado por aproximadamente 87,743 imágenes únicas acomodadas en 20,211 secuencias. El conjunto de datos brinda relatos y descripciones para las imágenes. Tanto las descripciones como los relatos fueron generados por personas a través de *Amazon Mechanical Turk*¹. Todas las imágenes en el conjunto de datos cuentan con descripción, sin embargo, solamente una proporción de las imágenes cuenta con relatos. Ejemplos de las imágenes, descripciones y relatos que se pueden encontrar en este conjunto de datos se presentan en el cuadro 5.1.

VIST está conformado por un conjunto de imágenes acomodadas en álbumes de distintos tamaños. De cada álbum, se crearon secuencias de 5 imágenes que representan eventos a partir de los cuales se puede contar una historia. Cada una de estas secuencias representa un relato único en el conjunto de datos. Estas secuencias fueron las utilizadas en este trabajo.

5.1.2. Recopilación

El conjunto de datos se encuentra libre para su descarga en [25] y se encuentra dividido en imágenes y texto. El texto a su vez está clasificado en narrativo y descriptivo. Para los fines de este trabajo, se utilizó solamente el texto narrativo, por lo tanto de aquí en adelante únicamente se detallará sobre este.

Las imágenes y los relatos están divididos en 3 conjuntos: entrenamiento, prueba y validación. Para cada uno de estos conjuntos se cuenta con un archivo Json que contiene los relatos.

Debido a que no todas las imágenes participan en los relatos, el conjunto de datos tuvo que ser filtrado. Lo anterior fue realizado también debido a que algunas imáge-

¹Plataforma de Amazon que sirve para realizar trabajos simples y de bajo precio unitario que requieren un nivel de inteligencia que una máquina no puede realizar. Permite que los usuarios registrados como trabajadores ("*turkers*") puedan realizar tareas disponibles a cambio de una recompensa económica impuesta por el solicitante de la tarea.

	Entrenamiento	Validación	Prueba
Original	167,528	21,048	21,075
Depurado	64,106	8,117	8,104

Cuadro 5.2: Cantidad de imágenes en VIST originalmente y después de realizar filtrado.

	Entrenamiento	Validación	Prueba
Original	40,155	4,990	5,055
Depurado	39,650	4,990	5,055

Cuadro 5.3: Cantidad de relatos en VIST originalmente y después de realizar filtrado.

nes estaban dañadas y debieron ser descartadas. La distribución de los datos quedó como se indica en los cuadros 5.2 y 5.3. Como se puede apreciar, muchas imágenes fueron descartadas pero muy pocos relatos se perdieron, todos pertenecientes solamente a los datos de entrenamiento.

Finalmente, los datos quedaron en tres archivos Json (entrenamiento, validación y prueba), todos con la misma estructura que se presenta a continuación:

- **relatos** - En esta sección se presentan los relatos para cada álbum. Se encuentran acomodadas por relato, donde cada uno consiste en 5 imágenes. Cada imagen contiene su oración correspondiente en el relato, el id que la identifica, el orden que ocupa en el relato, el id del relato y el id del álbum.
- **imágenes** - Aquí se presenta información relevante de cada una de las imágenes, como el id de la imagen, a qué álbum pertenece, el alto y ancho de cada imagen en píxeles, el nombre de la imagen y la fecha en la que la fotografía fue tomada.

5.2. Métricas de evaluación

En esta sección se explican en detalle las métricas utilizadas para evaluar la calidad de los relatos obtenidos por el método. Es importante destacar que ninguna de las métricas presentadas a continuación ha sido concebida para la evaluación de sistemas que generan relatos o historias de algún tipo. Pero debido a lo reciente que es la tarea, ninguna métrica ha sido creada para evaluar este tipo de sistemas y por lo tanto se hará uso de métricas que son utilizadas para evaluar tareas similares, como lo son la descripción de imágenes y la traducción automática. Asimismo, la métrica METEOR es recomendada en [54] como forma de evaluar relatos a partir de una secuencia de imágenes.

En la mayoría de estas métricas se utilizan los conceptos de n -grama, textos *hipótesis* y textos *referencia*. Todas las métricas evalúan textos *hipótesis* que son aquellos generados por un sistema contra textos *referencia* los cuales son tomados de un conjunto de datos y son generados por humanos. La evaluación de los textos hipótesis usualmente se realiza a nivel de n -gramas. Un n -grama es una subsecuencia de n elementos de una secuencia dada. Para ciertos valores de n se tienen nombres especiales, en el caso donde $n = 1$ se denomina *unigrama*, para $n = 2$ se denomina *bigrama* y para $n = 3$ se denomina *trigrama*.

5.2.1. BLEU

BLEU [44] es una métrica utilizada para la evaluación de la calidad de traducciones realizadas por sistemas de traducción automática.

BLEU se calcula a nivel de frases y halla la precisión de n -gramas que son comunes entre un texto hipótesis y uno o varios textos de referencia. Sin embargo, se utiliza una precisión modificada con el fin de solventar ciertas deficiencias en la métrica. La

puntuación de BLEU está definida como:

$$\text{BLEU} = PB \cdot \left(\sum_{n=1}^N w_n \cdot \log P_n \right)$$

Donde P_n es la precisión de n -gramas entre la descripción hipótesis y la descripción de referencia, w_n es el peso de cada n -grama (por lo general, $w_n = \frac{1}{N}$, pues $\sum_{n=1}^N w_n = 1$) y PB es el factor de penalización por brevedad definido como:

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

Donde h es la longitud del texto hipótesis y r la longitud del texto de referencia.

5.2.2. METEOR

METEOR [3] es también utilizada para evaluar sistemas de traducción automática. Mide la precisión y exhaustividad de unigramas, incluyendo además de los emparejamientos de palabras iguales, emparejamientos de palabras similares, basándose en sinónimos y raíces de palabras obtenidos de WordNet².

La puntuación de METEOR se basa en calcular dos medidas importantes: $Fmean$ y $Penalty$. El primer factor, $Fmean$ está definido como:

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

Aquí, P y R son la precisión y exhaustividad de unigramas respectivamente, ambas calculadas como:

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

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

²WordNet es una base de datos léxica en inglés. Agrupa palabras en conjuntos de sinónimos y proporciona definiciones cortas y generales, manteniendo las relaciones semánticas entre los conjuntos de sinónimos.

Donde m es el número de unigramas en el texto hipótesis que también aparecen en el texto de referencia, y w_t y w_r son el número de unigramas en el texto hipótesis y de referencia, respectivamente. De esta manera, F_{mean} se encarga de combinar la precisión y exhaustividad basados en emparejamientos de unigramas.

El concepto de penalización $Penalty$ es calculado mediante:

$$Penalty = 0,5 \cdot \frac{c}{u}$$

Donde c es el número de fragmentos de unigramas del texto hipótesis emparejados con el texto de referencia que se encuentran adyacentes y u es el número de unigramas emparejados. Finalmente, el puntaje METEOR está definido como:

$$METEOR = F_{mean} \cdot (1 - Penalty)$$

Para un solo texto hipótesis, METEOR calcula el puntaje respecto a cada uno de los textos de referencia y el mejor resultado es reportado.

5.2.3. CIDEr

CIDEr [57] fue creada especialmente para la evaluación de sistemas que realizan descripciones de imágenes. En general, utiliza el promedio de la distancia coseno entre los n -gramas del texto de referencia y los del texto hipótesis.

Sea $h_k(s_{ij})$ el número de veces que un n -grama w_k ocurre en el texto de referencia s_{ij} y similarmente $h_k(c_{ij})$ para el texto hipótesis c_{ij} , se calcula el TF-IDF (Term Frequency Inverse Document Frequency) $g_k(s_{ij})$ para cada n -grama w_k :

$$g_k(s_{ij}) = \frac{h_k(s_{ij})}{\sum_{w_l \in \Omega} h_l(s_{ij})} \cdot \log \beta$$

Donde Ω es el vocabulario de todos los n -gramas, y:

$$\beta = \left(\frac{|I|}{\sum_{I_p \in I} \min(1, \sum_q h_k(s_{pq}))} \right)$$

Donde I es el conjunto de imágenes y $\sum_q h_q(s_{pq})$ es el número de imágenes en las cuales w_k ocurre en alguno de los textos de referencia.

El puntaje CIDEr_n para n -gramas de longitud n se calcula utilizando la distancia coseno entre el texto hipótesis y los textos de referencia:

$$\text{CIDEr}_n(c_i, S_i) = \frac{1}{m} \sum_j \frac{g^n(c_i) \cdot g^n(s_{ij})}{\|g^n(c_i)\| \|g^n(s_{ij})\|}$$

Donde $g^n(c_i)$ es un vector formado por $g_k(c_i)$ que corresponde a todos los n -gramas de longitud n y $\|g^n(c_i)\|$ es la magnitud del vector. De manera similar se calcula para $g^n(s_{ij})$. La combinación de todos los n -gramas de distintas longitudes se realiza de la siguiente forma:

$$\text{CIDEr}(c_i, S_i) = \sum_{n=1}^N w_n \text{CIDEr}_n(c_i, S_i)$$

Los autores afirman que cuando $w_n = \frac{1}{N}$, se obtienen mejores resultados.

5.2.4. ROUGE-L

Un conjunto de métricas muy popular en sistemas que realizan resúmenes automáticos y traducción automática es ROUGE [37]. En este trabajo se utilizará una instancia de este conjunto llamada ROUGE-L. Esta métrica toma en cuenta la subsecuencia común más larga entre un texto de referencia y un texto hipótesis para calcular la puntuación.

Subsecuencia común más larga Una secuencia $Z = \{z_1, z_2, \dots, z_n\}$ es una subsecuencia de $X = \{x_1, x_2, \dots, x_m\}$ si existe estrictamente una secuencia incremental $[i_1, i_2, \dots, i_k]$ de índices en X tal que para todo $j = 1, 2, \dots, k$ se tiene $x_{i_j} = z_j$ [9]. Entonces, dadas dos secuencias X y Y , la subsecuencia común más larga (LCS) entre X y Y es aquella subsecuencia común de mayor longitud.

La intuición de utilizar LCS para la evaluación de un texto hipótesis respecto a un texto de referencia es que mientras más grande sea la LCS entre estos dos, más parecidos serán. Por lo tanto, ROUGE-L utiliza LCS para evaluar la similitud entre

un texto de referencia X de longitud m y un texto hipótesis Y de longitud n de la siguiente manera:

$$R_{lcs} = \frac{LCS(X, Y)}{m}$$

$$P_{lcs} = \frac{LCS(X, Y)}{n}$$

$$\text{ROUGE-L} = \frac{(1 + \beta^2) \cdot R_{lcs} \cdot P_{lcs}}{R_{lcs} + \beta^2 \cdot P_{lcs}}$$

Donde $LCS(X, Y)$ es la longitud de la subsecuencia común más larga entre X y Y y $\beta = P_{lcs}/R_{lcs}$.

5.3. Entrenamiento del método

La implementación del método se realizó haciendo uso del lenguaje de programación *Python* (versión 3.5.2) y la librería *Tensorflow* (versión 1.0). *Tensorflow* es una librería de código abierto creada por *Google* que fue concebida para la implementación eficiente de arquitecturas de aprendizaje profundo. El código fuente se puede encontrar en línea en el repositorio de *Github*³ [16].

El entrenamiento del método se llevó a cabo en un GPU *Titan X Maxwell* en un periodo de 40 días aproximadamente. Se entrenó con las 39,650 secuencias de imágenes, cada una con su respectivo relato, como se indica en el cuadro 5.3. El entrenamiento se realizó durante aproximadamente 120 épocas (500,000 pasos). La optimización se realizó sobre la función de error de entropía cruzada definida en la ecuación 4.5 de cada uno de los decodificadores haciendo uso del algoritmo de descenso por gradiente estocástico explicado en la sección 3.3.1.

³Plataforma de desarrollo colaborativo para alojar el código fuente de programas de computadora utilizando el sistema de control de versiones Git.

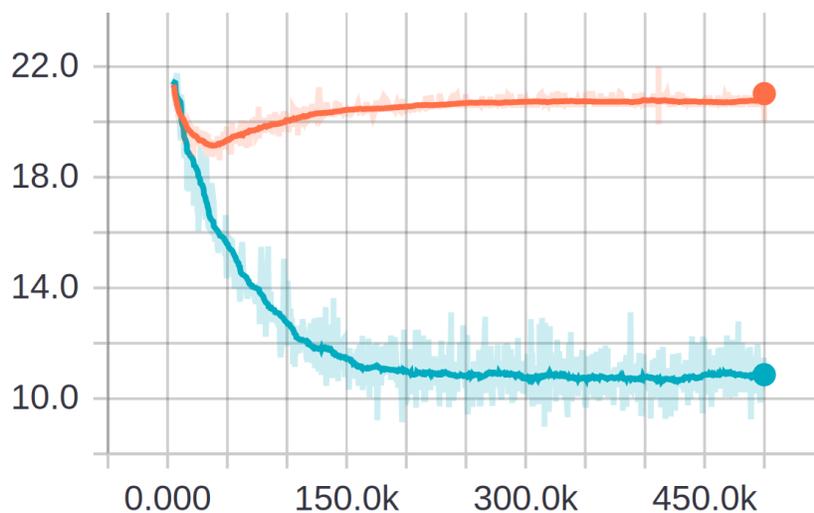


Figura 5.1: Perplejidad en los datos de entrenamiento y validación del método propuesto. Se muestra el valor de perplejidad en los datos de entrenamiento (azul) y en los datos de validación (naranja). En el eje horizontal se presenta cada *paso* del entrenamiento y el eje vertical indica el valor de perplejidad total. El valor de perplejidad total es la suma de la perplejidad en los 5 decodificadores individuales.

5.3.1. Perplejidad en modelos de lenguaje

Una manera común de evaluar modelos que generan lenguaje natural es calculando el valor de perplejidad sobre los datos de entrenamiento y prueba. En general, la perplejidad indica qué tan bien es que un modelo podrá predecir las palabras de un conjunto de prueba. En concreto, el valor de perplejidad está dado por:

$$PP = 2^{-\frac{1}{N} \sum_{i=1}^N \log_2 p(s_i)} \quad (5.1)$$

Donde N indica el número de palabras en el conjunto de datos y s_i indica la palabra a predecir. Un valor alto de perplejidad indicaría que el modelo asigna una gran probabilidad a todas las palabras del conjunto de datos. Lo anterior no es deseable ya que la aparición de todas las palabras no es igualmente probable. El peor valor de perplejidad que un modelo de lenguaje puede alcanzar es igual a la cardinalidad del vocabulario, por lo tanto, lo ideal es que el modelo logre un valor significativamente

más pequeño que este.

El desempeño del entrenamiento de la arquitectura se presenta en la figura 5.1. Este desempeño está representado mediante el valor de perplejidad calculado sobre los datos de entrenamiento y validación. Debido a que la arquitectura está compuesta por múltiples decodificadores, la gráfica en realidad representa la suma del valor de perplejidad de los cinco decodificadores. Como es de esperarse, el valor de perplejidad en los datos de entrenamiento tiende a bajar conforme el entrenamiento avanza. Sin embargo, en los datos de validación el valor de perplejidad permaneció estable y no tuvo una disminución considerable durante el entrenamiento.

5.4. Evaluación del método

5.4.1. *Visual Storytelling Challenge*

Con la finalidad de evaluar los relatos generados, se participó en el *Visual Storytelling Challenge* como parte del congreso *North American Chapter of the Association for Computational Linguistics* del 2018. El concurso fue organizado por los autores de [54] y es el primero que se realiza para esta tarea en particular. La tarea a resolver en concreto consistía en la generación de relatos a partir de secuencias de 5 imágenes haciendo uso del conjunto de datos VIST.

El concurso consistió en dos categorías: *Internal Track* y *External Track*. La primer categoría fue para sistemas que solamente utilizaban el conjunto de datos VIST para su entrenamiento. Por otra parte, la segunda categoría fue para sistemas que utilizaban otros conjuntos de datos además del conjunto de datos VIST. La arquitectura propuesta en este trabajo no utiliza otro conjunto de datos adicional para su entrenamiento y por esto se participó solamente en la primer categoría.

Solamente se contó con la participación de 4 equipos en la categoría *Internal Track* y ningún equipo en la categoría *External Track*. Los 4 equipos participantes fueron: NLPSA501 [23], UCSB-NLP [61], SnuBiVtt [28] y DG-DLMX. Este último es

	Conjunto de prueba público	Conjunto de prueba secreto
Método propuesto	0.3088	0.3100
SnuBiVtt [28]	0.2982	0.3066
UCSB-NLP [61]	0.3179	0.317
NLPSA501 [23]	0.3171	0.3184

Cuadro 5.4: Resultados de la evaluación con METEOR de los relatos generados por el método propuesto.

el nombre del método presentado en este trabajo.

La evaluación de los relatos sometidos por los equipos se realizó mediante la métrica automática METEOR y mediante evaluación humana. Sin embargo, para la elección del equipo ganador se tomó en cuenta solamente las puntuaciones obtenidas en la evaluación humana. A continuación se detallan los resultados obtenidos en el concurso de acuerdo a cada una de estas evaluaciones.

Evaluación automática

Para la evaluación automática se tomaron dos conjuntos de datos diferentes y se calculó la métrica METEOR sobre cada uno. El primero fue un conjunto de prueba público el cual consistió en 1,938 relatos elegidos aleatoriamente del conjunto de prueba de VIST y un conjunto de prueba secreto conformado por un subconjunto desconocido del conjunto de prueba público, elegido por los organizadores del concurso. En el cuadro 5.4 se presentan los resultados obtenidos por el método propuesto en ambos conjuntos.

Evaluación humana

La evaluación humana fue realizada mediante *Amazon Mechanical Turk* en la que participaron 5 personas para calificar los relatos generados por los 4 sistemas. Entre los relatos a evaluar se agregaron relatos generados por humanos para utilizar estos

	a)	b)	c)	d)	e)	f)	Total
Humanos	4.025	3.975	3.772	4.003	3.965	3.857	23.596
Método propuesto	3.347	3.278	2.871	3.222	2.886	2.893	18.498
SnuBiVtt [28]	3.548	3.524	3.075	3.589	3.236	3.323	20.295 ⁵
UCSB-NLP [61]	3.236	3.065	2.767	3.029	3.032	2.867	17.995
NLPSA501 [23]	3.111	2.870	2.769	2.870	3.072	2.881	17.574

Cuadro 5.5: Evaluación humana de los relatos generados por los equipos de la competencia comparados contra relatos generados por humanos. Los aspectos evaluados son: a) ¿el relato es enfocado?, b) estructura y coherencia del relato, c) ¿compartirías el relato?, d) ¿el relato fue escrito por un humano?, e) ¿el relato está relacionado con las imágenes?, f) ¿qué tan detallado es el relato?

puntajes como referencia. En esta evaluación solamente se eligieron 200 relatos para ser evaluados, estos fueron tomados del conjunto de prueba secreto utilizado en la evaluación automática. Para calificar los relatos, se utilizó una escala Likert⁴ y los evaluadores debían considerar los siguientes 6 aspectos:

- a) Enfocada: cada oración del relato debe contener información que sea “naturalmente” relevante para el resto del relato, nada debe ser aleatorio o irrelevante.
- b) Estructura y coherencia: el relato debe estar bien estructurado, ser correcto gramaticalmente y estar bien organizado.
- c) ¿Compartirías este relato?: si estas fueran tus imágenes, ¿compartirías este relato con ellas?
- d) ¿Fue escrito por un humano?: este relato parece ser escrito por una persona.

⁴Es la escala de uso más amplio en encuestas para la investigación. Con esta escala se especifica el nivel de acuerdo o desacuerdo con una declaración, elemento o pregunta.

⁵Sistema presentado después de la fecha límite establecida del concurso. Los organizadores acordaron incluir el resultado del sistema en la evaluación, pero no considerarlo para elegir el ganador de la competencia.

	BLEU-4	METEOR	ROUGE-L	CIDEr
Huang et al. [54]	-	31.4	-	-
Kim et al. [45]	-	30.63	-	-
Yu et al. [64]	-	34.1	29.5	7.5
UCSB-NLP [61]	14.1	35.2	29.6	9.5
Método propuesto	12.7	34.4	29.2	5.1

Cuadro 5.6: Evaluación con métricas de los relatos generados por el método propuesto en el conjunto de prueba de VIST.

- e) Relacionado a las imágenes: este relato habla sobre entidades concretas en las fotos.
- f) Detallada: este relato proporciona un nivel apropiado de detalle.

En el cuadro 5.5 se presentan los resultados obtenidos por el modelo propuesto en la evaluación humana.

5.4.2. Evaluación con métricas

Por último, se realizó una evaluación sobre todo el conjunto de prueba de VIST que consistió de 5,055 secuencias de imágenes. Los relatos generados para estas secuencias fueron evaluados haciendo uso de las métricas automáticas presentadas en la sección 5.2. En el cuadro 5.6 se presentan los resultados obtenidos para el conjunto de prueba. La figura 5.2 presenta algunos ejemplos de los relatos generados para la competencia. Asimismo, en el anexo A se pueden encontrar más ejemplos de relatos generados para secuencias de imágenes extraídas del conjunto de prueba de VIST.

5.5. Discusión

En general, los resultados obtenidos en todas las evaluaciones fueron bastante competitivos. Se obtuvieron buenas puntuaciones con las métricas automáticas en

ambos conjuntos de prueba de la competencia, así como en todo el conjunto de prueba de VIST. En cuanto a la evaluación humana, los relatos generados por el método propuesto obtuvieron resultados favorables lo cual se vio reflejado en haber obtenido el primer lugar en la categoría *Internal Track* de la competencia *Visual Storytelling Challenge*.

En general, los relatos generados por el método propuesto son gramaticalmente correctos y coherentes pero la mayoría de ellos tienden a contener ideas o frases repetitivas y muy generales. Lo anterior se puede comprobar en la evaluación humana ya que uno de los mejores aspectos calificados fue la estructura y coherencia de los relatos y uno de los peores fue el nivel de detalle.

Como se muestra en la figura 5.2, algunos de los relatos generados contienen errores pequeños y si se compara de manera *frase-imagen* la frase puede no estar relacionada con el contenido real de la imagen, un ejemplo de esto se encuentra en la tercer frase del primer relato o en la cuarta frase del último relato en la figura 5.2. Esto se vio reflejado en uno de los aspectos con más baja puntuación, la relación de los relatos con el contenido de las imágenes. Sin embargo, si la comparación se realizara de manera *secuencia-relato* puede verse más relacionado el contenido del relato con el contenido de las imágenes. Lo anterior puede ser consecuencia de que los decodificadores del método son alimentados con información tanto global como individual y que esto ocasione que la información global (de la secuencia) prevalezca sobre la información particular (de la imagen en cuestión) y por lo tanto, la frase generada por el decodificador esté más relacionada a la esencia del álbum que a la imagen particular.

Por otro lado, la evaluación humana refleja que los relatos generados son muy generales y no presentan tantos detalles, un ejemplo de esto se puede ver en las últimas frases del tercer relato en la figura 5.2. Es altamente probable que estos dos factores tuvieran un impacto directo en el hecho que las personas no deseen compartir los relatos.



"We went to the city today. There were a lot of people there. The food was delicious. There was also a lot of beautiful flowers. We had a great time."



"We went to the beach today. We had a great time playing the water. The view of the water was amazing. The beach was beautiful. After a long day of swimming, the family decided to take a break."



"The boys were excited for their first day of school. He was so excited to see everyone. We had a great time. The kids had a great time. We had a great time."



"I went to the party last night. We had a lot of fun taking pictures together. We all had a great time together. The girls had a good time. The night ended with a great band."

Figura 5.2: Ejemplos de relatos generados por el método propuesto.

6

Conclusiones y trabajo futuro

La tarea de generar relatos a partir de una secuencia de imágenes es muy compleja y reciente, y por lo tanto los modelos presentados en este trabajo son los primeros intentos de resolver este problema.

En este trabajo de investigación se propuso una arquitectura para la generación de relatos a partir de una secuencia de cinco imágenes. Esta arquitectura está compuesta por un codificador que procesa la secuencia de imágenes y obtiene un vector de contexto, el cual es utilizado por 5 decodificadores independientes para generar partes del relato completo. El estado oculto inicial de cada decodificador es inicializado con el vector de contexto obtenido por el codificador y recibe como primer entrada la imagen correspondiente de la secuencia. Cada decodificador genera una secuencia de palabras que es una parte del relato completo. El relato final es la unión de todas estas partes.

Los relatos generados por el método propuesto son correctos gramatical y estructuralmente. El uso de múltiples decodificadores que utilizan información global e individual permitió generar relatos informados sobre el contenido de todas las imágenes en la secuencia. Asimismo, el uso de decodificadores independientes para cada posición de la secuencia permitió construir modelos de lenguaje más específicos que ayudaron en la calidad de los relatos generados.

El método propuesto tuvo un buen desempeño al ser evaluado con métricas automáticas y en la evaluación humana. Lo anterior se manifestó en los resultados obtenidos en la competencia *Visual Storytelling Challenge* de la *North American Chapter of the Association for Computational Linguistics* del 2018.

Del desempeño de la arquitectura propuesta en este trabajo se pueden considerar los siguientes puntos como posible trabajo futuro, en busca de solventar los puntos débiles y mejorar la calidad de los relatos generados:

- **Reemplazar *Inception V3*** - Cambiar la red para la obtención de la representación de las imágenes por un modelo más robusto como *DenseNet* [24] que permita capturar más información y de mejor calidad. Lo anterior es propuesto para contar con mejor información sobre el contenido de las imágenes en la secuencia que permita generar relatos más relacionados a las imágenes.
- **Reemplazar *Word2vec*** - Utilizar otra forma de representación del texto como *fastText* [5]. Esta librería es el estado del arte en modelos que trabajan con representaciones vectoriales de lenguaje natural, como por ejemplo modelos para clasificación de texto los cuales son utilizados en aplicaciones de generación de respuestas inteligentes y análisis de sentimiento, por nombrar algunas.
- **Mecanismo de atención** - Implementar un mecanismo de atención en los decodificadores de la arquitectura para solucionar los problemas de ideas o frases repetitivas dentro de un mismo relato. Esto también podría ayudar a generar relatos más precisos que no contengan ideas tan generales como se pudo observar en algunos de los resultados obtenidos.

El mecanismo de atención permitiría a cada decodificador atender ciertas partes de la oración de entrada en cada instante en que genera una palabra de la oración de salida. De esta manera el decodificador aprende las partes más importantes a las que debe prestar atención para generar un buen resultado.

- **Otra arquitectura** - Una de las principales desventajas de utilizar un método

secuencia a secuencia como el utilizado en este trabajo, es que se espera que el decodificador genere una salida solamente a partir del contexto obtenido del codificador. Lo anterior se vuelve un problema al lidiar con oraciones muy largas ya que al contar con entradas muy grandes se necesita que el codificador resuma mucha información en un solo vector de tamaño fijo. Un enfoque alternativo puede ser considerar utilizar una arquitectura basada en mecanismos de atención. Un ejemplo de esto es el modelo conocido como *Transformer* [56], en donde no se almacena información en un solo vector si no se presta atención solamente a las partes más relevantes de la oración de entrada para generar una salida. Con esta arquitectura se han obtenido resultados destacados en tareas de modelado de lenguaje y es considerada el estado del arte en problemas relacionados a la comprensión del lenguaje natural.

Bibliografía

- [1] AGRAWAL, A., CHANDRASEKARAN, A., BATRA, D., PARIKH, D., AND BANSAL, M. Sort story: Sorting jumbled images and captions into stories. In *Empirical Methods in Natural Language Processing* (2016).
- [2] ANG, K., YU, S., AND ONG, E. Theme-based cause-effect planning for multiple-scene story generation. In *Proceedings of the Second International Conference on Computational Creativity* (2011), pp. 48–53.
- [3] BANERJEE, S., AND LAVIE, A. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization* (2005).
- [4] BENGIO, Y., SIMARD, P., AND FRASCONI, P. Learning long term dependencies with gradient descent is difficult. In *IEEE Transactions on Neural Networks* (1994).
- [5] BOJANOWSKI, P., GRAVE, E., JOULIN, A., AND MIKOLOV, T. Enriching word vectors with subword information. In *Transactions of the Association for Computational Linguistics* (2017), pp. 135–146.
- [6] CAUCHY, A. Méthode générale pour la résolution de systèmes d'équations simultanées. In *Proceedings of the Academy of sciences* (1847).

- [7] CHO, K., MERRIËNBOER, B., GULCEHRE, C., BOUGARES, F., BAHDANAU, D., SCHWENK, H., AND BENGIO, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the Empirical Methods in Natural Language Processing* (2014).
- [8] CHO, K., VAN MERRIENBOER, B., BAHDANAU, D., AND BENGIO, Y. On the properties of neural machine translation: Encoder-decoder approaches. *ArXiv e-prints* (2014).
- [9] CORMEN, T. R., LEISERSON, C. E., AND RIVEST, R. L. *Introduction to Algorithms*. The MIT Press, 1989.
- [10] DONAHUE, J., HENDRICKS, L. A., ROHRBACH, M., VENUGOPALAN, S., GUADARRAMA, S., SAENKO, K., AND DARRELL, T. Long-term recurrent convolutional networks for visual recognition and description. In *Computer Vision and Pattern Recognition* (2015).
- [11] ELMAN, J. Finding structure in time. In *Cognitive Science* (1990), pp. 179–211.
- [12] FAN, C., AND CRANDALL, D. J. DeepDiary: Automatic caption generation for lifelogging image streams. *ArXiv e-prints* (2016).
- [13] FARHADI, A., HEJRATI, M., SADEGHI, M. A., YOUNG, P., RASHTCHIAN, C., HOCKENMAIER, J., AND FORSYTH, D. Every picture tells a story: Generating sentences from images. In *European Conference on Computer Vision* (2010).
- [14] GERS, F. A., AND SCHMIDHUBER, J. Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS* (2000).
- [15] GÓMEZ DE SILVA GARZA, A., AND PÉREZ Y PÉREZ, R. Towards evolutionary story generation. In *International Conference on Computational Creativity* (2014).
- [16] GONZÁLEZ-RICO, D. Contextualize, show and tell: A neural visual storyteller. <https://github.com/dgonzalez-ri>, 2018.

-
- [17] GOODFELLOW, I., BENGIO, Y., AND COURVILLE, A. *Deep Learning*. MIT Press, 2016.
- [18] GREFF, K., SRIVASTAVA, R. K., KOUTNÍK, J., STEUNEBRINK, B. R., AND SCHMIDHUBER, J. Lstm: A search space odyssey. In *IEEE Transactions on Neural Networks and Learning Systems* (2015).
- [19] GUO, Y., YAO, B., AND LIU, Y. Sequence to sequence model for video captioning.
- [20] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition* (2016).
- [21] HOCHREITER, S., AND SCHMIDHUBER, J. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
- [22] HONG, A., SOLIS, C. J., SIY, J. T., TABIRAO, E., AND ONG, E. Planning author and character goals for story generation. In *Proceedings of the NAACL Human Language Technology Workshop on Computational Approaches to Linguistic Creativity* (2009), pp. 63–70.
- [23] HSU, C., CHEN, S., HSIEH, M., AND KU, L. Using inter-sentence diverse beam search to reduce redundancy in visual storytelling. *CoRR abs/1805.11867* (2018).
- [24] HUANG, G., LIU, Z., VAN DER MAATEN, L., AND WEINBERGER, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017).
- [25] HUANG, T. K., FERRARO, F., MOSTAFAZADEH, N., MISRA, I., AGRAWAL, A., DEVLIN, J., GIRSHICK, R., HE, X., KOHLI, P., BATRA, D., ZITNICK, C. L., PARIKH, D., VANDERWENDE, L., GALLEY, M., AND MITCHELL, M. Visual storytelling dataset. <http://www.visionandlanguage.net/VIST/>, 2016. [Online; accessed 19-04-2017].

-
- [26] JOZEFOWICZ, R., ZAREMBA, W., AND SUTSKEVER, I. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning* (2015).
- [27] KARPATHY, A., AND LI, F. Deep visual-semantic alignments for generating image descriptions. In *Computer Vision and Pattern Recognition* (2015).
- [28] KIM, T., HEO, M., SON, S., PARK, K., AND ZHANG, B. GLAC net: Glocal attention cascading networks for multi-image cued story generation. *CoRR abs/1805.10973* (2018).
- [29] KINGMA, D. P., AND BA, J. Adam: A method for stochastic optimization. *CoRR abs/1412.6980* (2014).
- [30] KIROS, R., SALAKHUTDINOV, R., AND ZEMEL, R. Multimodal neural language models. In *Proceedings of the 31st International Conference on Machine Learning* (2014), pp. 595–603.
- [31] KIROS, R., ZHU, Y., SALAKHUTDINOV, R., ZEMEL, R. S., TORRALBA, A., URTASUN, R., AND FIDLER, S. Skip-thought vectors. *ArXiv e-prints* (2015).
- [32] KLEIN, S., AESCHLIMANN, J., BALSIGER, D., CONVERSE, S., COURT, C., FOSTER, M., LAO, R., OAKLEY, J., AND SMITH, J. Automatic novel writing: A status report. *Computer Science Department, The University of Wisconsin* (1973).
- [33] KRASIN, I., DUERIG, T., ALLDRIN, N., VEIT, A., ABU-EL-HAJA, S., BELONGIE, S., CAI, D., FENG, Z., FERRARI, V., GOMES, V., GUPTA, A., NARAYANAN, D., SUN, C., CHECHIK, G., AND MURPHY, K. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from <https://github.com/openimages>* (2016).
- [34] KRIZHEVSKY, A. Learning multiple layers of features from tiny images.

-
- [35] KULKARNI, G., PREMRAJ, V., DHAR, S., LI, S., CHOI, Y., BERG, A. C., AND BERG, T. L. Baby talk: Understanding and generating simple image descriptions. In *Computer Vision and Pattern Recognition* (2011).
- [36] LECUN, Y., BOTTOU, L., BENGIO, Y., AND HAFFNER, P. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE* (1998).
- [37] LIN, C.-Y. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop* (2004).
- [38] LIN, T.-Y., MAIRE, M., BELONGIE, S., BOURDEV, L., GIRSHICK, R., HAYS, J., PERONA, P., RAMANAN, D., ZITNICK, C. L., AND DOLLAR, P. Microsoft COCO: Common objects in context. *ArXiv e-prints* (2014).
- [39] LIU, J., FU, J., MEI, T., AND CHEN, C. W. Let your photos talk: Generating narrative paragraph for photo stream via bidirectional attention recurrent neural networks. In *Association for the Advancement of Artificial Intelligence* (2017).
- [40] MAO, J., XU, W., YANG, Y., WANG, J., HUANG, Z., AND YUILLE, A. L. Deep captioning with multimodal recurrent neural networks (m-rnn). In *International Conference on Learning Representations* (2015), pp. 595–603.
- [41] MAO, J., XU, W., YANG, Y., WANG, J., AND YUILLE, A. L. Explain images with multimodal recurrent neural networks. *ArXiv e-prints* (2014).
- [42] MIKOLOV, T., CHEN, K., CORRADO, G., AND DEAN, J. Efficient estimation of word representations in vector space. *ArXiv e-prints* (2013).
- [43] MITCHELL, T. M. *Machine Learning*. McGraw-Hill, 1997.
- [44] PAPANENI, K., ROUKOS, S., WARD, T., AND ZHU, W. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (2002).

-
- [45] PARK, C. C., AND KIM, G. Expressing an image stream with a sequence of natural sentences. In *Advances in Neural Information Processing Systems 28*. 2015, pp. 73–81.
- [46] PÉREZ Y PÉREZ, R. *MEXICA: A computer model of creativity in writing*. PhD thesis, Universidad de Sussex, 1999.
- [47] RUMELHART, D., HINTON, G., AND WILLIAMS, R. Learning representations by back-propagating errors. *Nature* (1986).
- [48] SAMIM. Generating stories about images – recurrent neural network for generating stories about images. <https://medium.com/@samim/generating-stories-about-images-d163ba41e4ed>, 2015. [Online; accessed 04-Jul-2018].
- [49] SHI, X., CHEN, Z., WANG, H., YEUNG, D., WONG, W., AND WOO, W. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems* (2015).
- [50] SINGH, D., ACKERMAN, M., AND PÉREZ Y PÉREZ, R. A ballad of the mexicas: Automated lyrical narrative writing. In *International Conference on Computational Creativity* (2017).
- [51] SUN, C., SHRIVASTAVA, A., SINGH, S., AND GUPTA, A. Revisiting unreasonable effectiveness of data in deep learning era. *CoRR abs/1707.02968* (2017).
- [52] SUTSKEVER, I., VINYALS, O., AND LE, Q. V. Sequence to sequence learning with neural networks. In *Neural Information Processing Systems* (2014).
- [53] SZEGEDY, C., VANHOUCHE, V., IOFFE, S., SHLENS, J., AND WOJNA, Z. Rethinking the Inception architecture for computer vision. *ArXiv e-prints* (2015).
- [54] TING-HAO, HUANG, FERRARO, F., MOSTAFAZADEH, N., MISRA, I., AGRAWAL, A., DEVLIN, J., GIRSHICK, R., HE, X., KOHLI, P., BATRA, D., ZITNICK, C. L., PARIKH,

- D., VANDERWENDE, L., GALLEY, M., AND MITCHELL, M. Visual storytelling. In *North American Conference on Chinese Linguistics* (2016).
- [55] TRAN, K., HE, X., ZHANG, L., SUN, J., CARAPCEA, C., THRASHER, C., BUEHLER, C., AND SIENKIEWICZ, C. Rich image captioning in the wild. *CoRR abs/1603.09016* (2016).
- [56] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L., AND POLOSUKHIN, I. Attention is all you need. *ArXiv e-prints* (2017).
- [57] VEDANTAM, R., ZITNICK, C. L., AND PARIKH, D. CIDEr: Consensus-based Image Description Evaluation. *ArXiv e-prints* (2014).
- [58] VENOUR, C., AND REITER, E. A tutorial for simplenlg. <http://www.csd.abdn.ac.uk/~ereiter/simplenlg>, 2008. [Online; accessed 19-04-2017].
- [59] VENUGOPALAN, S., ROHRBACH, M., DONAHUE, J., MOONEY, R., DARRELL, T., AND SAENKO, K. Sequence to sequence – video to text. *ArXiv e-prints* (2015).
- [60] VINYALS, O., TOSHEV, A., BENGIO, S., AND ERHAN, D. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition* (2015).
- [61] WANG, X., CHEN, W., WANG, Y.-F., AND WANG, W. Y. No metrics are perfect: Adversarial reward learning for visual storytelling. *ArXiv e-prints* (2018).
- [62] WU, Y., SCHUSTER, M., CHEN, Z., LE, Q. V., NOROUZI, M., MACHEREY, W., KRIKUN, M., CAO, Y., GAO, Q., MACHEREY, K., KLINGNER, J., SHAH, A., JOHNSON, M., LIU, X., KAISER, L., GOUWS, S., KATO, Y., KUDO, T., KAZAWA, H., STEVENS, K., KURIAN, G., PATIL, N., WANG, W., YOUNG, C., SMITH, J., RIESA, J., RUDNICK, A., VINYALS, O., CORRADO, G., HUGHES, M., AND DEAN, J. Google’s neural machine translation system: Bridging the gap between human and machine translation. *CoRR abs/1609.08144* (2016).

- [63] WU, Y., SCHUSTER, M., CHEN, Z., LE, Q. V., NOROUZI, M., MACHEREY, W., KRIKUN, M., CAO, Y., GAO, Q., MACHEREY, K., KLINGNER, J., SHAH, A., JOHNSON, M., LIU, X., KAISER, Ł., GOUWS, S., KATO, Y., KUDO, T., KAZAWA, H., STEVENS, K., KURIAN, G., PATIL, N., WANG, W., YOUNG, C., SMITH, J., RIESA, J., RUDNICK, A., VINYALS, O., CORRADO, G., HUGHES, M., AND DEAN, J. Google’s neural machine translation system: Bridging the gap between human and machine translation. *ArXiv e-prints* (2016).
- [64] YU, L., BANSAL, M., AND BERG, T. L. Hierarchically-Attentive RNN for album summarization and storytelling. *ArXiv e-prints* (2017).
- [65] ZHU, Y., KIROS, R., ZEMEL, R., SALAKHUTDINOV, R., URTASUN, R., TORRALBA, A., AND FIDLER, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *ArXiv e-prints* (2015).

Anexo A

Relatos generados

A continuación se presentan más ejemplos de relatos generados por el método propuesto. Las secuencias de imágenes y relatos originales fueron obtenidos del conjunto de prueba de la base de datos VIST. Primero se muestra el relato original y después el relato generado por el modelo.



"The Fedora team got together at Saxby's for casual discussion. Many things were discussed in a nice and relaxed setting. It didn't seem like work in such a cozy place. There were many smiles. The meeting was a success with much done."

"I went to the party last night. We had a lot of fun. We all had a great time together. The girls had a great time dancing. It was a great night."



"James is leaving, and this cake commemorates the event honoring him. A colleague enjoys the cake and some conversation. Others are all smiles as they celebrate James and their time with him. More co-workers pose for a photo as they say goodbye to James. Here's James, waving goodbye as he leaves for the final time and new adventures await him."

"I made a cake for the party today. The food was delicious. We had a great time talking and laughing. Everyone was very happy. We had a lot of fun."



"Me and a few coworkers decided to go out for happy hour. My assistant was thrilled because she was tired from working so much and needed some relaxation. To my fellow coworkers couldn't wait to sit at the bar and have drinks. My two bosses were really enjoying themselves. It started getting late and we realized we didn't even have dinner yet."

"It was a great day to go to the beach. We took a selfie in front of our friends. We took a lot of pictures together. We had a lot of fun. The view from the city was amazing."



"Today is the most important day of my life, being married to my significant other. I decided to take a picture of the wedding site before everyone came and took their seats. The marriage certificate makes everything official however, it doesn't hold much value to me. My family decided to visit my house after the ceremony and reception. Everyone was too energetic and forgot about sleeping."

"I was so excited to be at the wedding. The bride and groom were very happy together. They had a lot of instructions for everyone. We all had a good time talking to each other. We had a great time."



"We see this man in a hospital gown. Everyone at the table seemed to enjoy the company of the man. He showed us his elaborate details of his costume. With food in their stomachs, they decided to take a break outside. Lastly, this couple won the costume contest."

"The couple was excited. We had a lot of fun together. The kids were having a blast. We all had a great time dancing. The night ended with a bond fire."



“Lucy was driving up north to visit her parents. Though it wasn't the drive she loved. She would take I-80 and eventually merge on to I-95 north. Lucy looked up at the sky.

It looked like rain was coming. She hated driving in the rain. Lucy saw many big trucks going the opposite way. It must be a big day for deliveries she thought, as she continued her journey. Lucy thought to herself, thank you lord, the skies have cleared up. It didn't look like rain anymore, but she still had four more hours of driving before she'd reach her parents house in Michigan.”

“I went on a road trip last weekend. It was a long drive out of the city. I had a great time. There were a lot of cars that day. We had a great time and can't wait to come back.”



“The Boy Scout troop got to tour the local Air Force base. The boys loved seeing the fighter jets. They got to watch as one of them was serviced. Everyone liked seeing the test dummy. The big hangar was beautiful. I hope the boys get to visit again next year.”

“I went to the museum. There were a lot of cool cars there. This is a picture of a man. This is the <UNK> of the <UNK>. This was the last one we saw before we left.”