Universitat Oberta de Catalunya

uoc.edu

Descripción automática de imágenes

Xavi Medina Torregrosa
Máster en Ciencia de datos
Área 1: Deep Learning

Anna Rue
Albert Solé Ribalta
08/01/2020

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Descripción automática de imágenes*
Nombre del autor:	*Xavi Medina Torregrosa*
Nombre del consultor/a:	*Anna Rue*
Nombre del PRA:	*Albert Solé Ribalta*
Fecha de entrega (mm/aaaa):	01/2020
Titulación:	*Máster en Ciencia de Datos*
Área del Trabajo Final:	*Área 1: Subárea 1.2: Deep Learning*
Idioma del trabajo:	*Castellano*
Palabras clave	*Image recognition, natural language processing, automatic description generation*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Este trabajo de final de máster consiste en poder generar de forma automática descripciones para imágenes. El alcance de este proyecto es generar descripciones que contengan información de los objetos de una imagen y sus atributos, es decir, no entraré en describir acciones o emociones de personas u objetos que puedan aparecer en una imagen.

El contexto de la aplicación del trabajo serán imágenes de productos, de manera que el objetivo final sea obtener un modelo capaz de generar las descripciones de estos productos teniendo en cuenta sus atributos principales.

Para lograr esto, lo primero será generar un conjunto de datos con todos los productos a tener en cuenta y sus variantes, para después crear un modelo capaz de identificar que productos se encuentran en cada imagen y sus características para poder generar una descripción con lenguaje natural capaz de ser interpretada por un humano después.

El resultado final del proyecto será tener un servicio que devuelva una descripción en inglés de la imagen pasada como parámetro de entrada, esto puede ser utilizado para generar catálogos de marketing para plataformas como Facebook, Amazon, Google etc. De manera automática cuando la empresa tiene productos configurables, o incluso para poder hacer una equivalencia entre productos de la competencia y de la propia empresa a partir de la imagen.

Abstract (in English, 250 words or less):

This master's final project consists in generating description automatically given an image. The scope of this project is to generate description that contain information of the objects in an image and their attributes, which means that this project won't focus on being able to describe actions or emotions from people or objects appearing in the image.

The context for this project will be product images, like the ones you would find in an e-commerce catalogue, being the final objective having a model able to generate the description for those products taking into account their attributes.

To achieve this, the first step will be generating the data set with all the products I want to take account and their variations, to later create the model able to identify those products in each image and their attributes to generate a description with natural language that can be understood by a human.

The final result of this project, will be having a service that returns a description in English of the image that was sent as input. This can be used to automatically generate marketing catalogues for third parties like Facebook, Amazon, Google, etc. when the company has a lot of configurable products and generating them manually can be difficult or costly. It could also be used when scraping competence data to match their products with yours in order to adjust prices.

Índice

1. Introducción 1
1.1 Contexto y justificación del Trabajo 1
1.2 Objetivos del Trabajo 1
1.3 Enfoque y método seguido 2
1.4 Planificación del Trabajo 2
1.5 Breve sumario de productos obtenidos 3
1.6 Breve descripción de los otros capítulos de la memoria 3
2. Estado del arte 4
3. Desarrollo del proyecto 5
3.1. Creación del conjunto de datos 5
3.2. Desarrollo del modelo predictivo 5
4. Conclusiones 12
5. Glosario 13
6. Bibliografía 14
7. Anexos 16
6.1. Repositorio del código y datos 16
6.2. Ejecución en el cloud 16

Lista de figuras

Figura 1. Diagrama Gantt del proyecto 3
Figura 2. Pipeline de generación de imágenes de Microsoft 4
Figura 3. Primera iteración de la CNN 6
Figura 4. Arquitectura VGG16 6
Figura 5. Modelo de clasificación final 7
Figura 6. Test del modelo de clasificación en una imagen de una taza 7
Figura 7. Diagrama del modelo de predicción de descripciones en primera fase 8
Figura 8. Arquitectura del modelo Inception V3 de Google 8
Figura 9. Modelo de preddicion de descripciones con Inception v3 y GloVe 9
Figura 10. Predicción de la descripción de la imagen correcta 9
Figura 11. Modelo conceptual de predicción de texto alterada por atributos ... 10
Figura 12. Pipeline final para la generación de descripciones 11

1. Introducción

1.1 Contexto y justificación del Trabajo

La idea de este trabajo surge de varias necesidades que he ido observando a lo largo de mi vida profesional al trabajar en varios ‘eCommerce’, especialmente cuando hay una gran cantidad de productos configurables que hacen que publicitar todas las opciones o personalizar el contenido para los usuarios sea un trabajo arduo de cubrir manualmente, de manera que poder generar el contenido del catálogo de manera automática puede agilizar este tipo de procesos y habilitar muchos otros casos de uso.

Actualmente estos catálogos se acaban creando de manera manual o reutilizando el contenido de un producto base para el resto de sus configuraciones posibles, lo que hace difícil crear un contenido atractivo y más personalizable a la hora de publicitarlo.

Con este proyecto quiero conseguir habilitar múltiples casos de uso, uno de ellos siendo el generar de manera automática los atributos de este tipo de catálogos introduciendo solamente las imágenes de los productos que queremos incluir, otro caso de uso es de poder identificar productos de otras compañías de la competencia y poderlos mapear a los nuestros, de manera que no necesitemos intervención humana.

Como parte de la generación de la descripción, también se pueden identificar ‘keywords’ que se pueden usar para mejorar los motores de búsqueda y posicionar mejor los productos.

En mi empresa actual, los usuarios pueden subir imágenes propias para personalizar nuestros productos, de manera que con la práctica y los conocimientos obtenidos de este proyecto, se pueden hacer variaciones similar como por ejemplo obtener información del tipo de diseños que le gusta a cada usuario y poder hacer un marketing más personalizable a ese nivel.

Por lo que el estado final de este proyecto ha de ser un servicio al que enviándole una imagen, nos devuelva una descripción generada automáticamente.

1.2 Objetivos del Trabajo

El objetivo final a conseguir es el de poder generar una descripción de un producto que contenga el tipo de producto que es y sus atributos

principales. Por ejemplo, si tenemos una imagen de una camiseta, poder generar una descripción que contenga el producto, el color, la tela, manga larga o corta, etc.

Como objetivos parciales para llegar a ese resultado final, podemos identificar los siguientes:

- Identificar cual es el actual procedimiento para proyectos similares en la industria.
- Crear el conjunto de datos necesario para entrenar los modelos necesarios.
- Identificar el producto que se encuentra en una imagen.
- Identificar los atributos de un producto en una imagen.
- Generar descripciones en ingles que contengan información del producto y sus atributos.
- Exponer un servicio que reciba una imagen y devuelva la descripción generada por el modelo.

1.3 Enfoque y método seguido

En cuanto al enfoque de este proyecto, seguiré una metodología cuantitativa para la investigación, ya que tengo acceso al contexto real donde los datos están y se utilizan y puedo recolectar los datos de manera fácil, hacer pruebas, y ver como se comportan los resultados para observar si son suficientemente buenos en el entorno real de estos.

En cuanto a las herramientas a utilizar, utilizaré Python haciendo uso de la librería Keras para hacer uso de TensorFlow más fácilmente.

En cuanto al trabajo en sí, crearé un nuevo modelo que pueda ser ejecutado desde un servicio publicado en la nube, para esto, la metodología de trabajo que seguiré será un modelo ágil, en el que comenzaré creando una prueba de concepto del modelo o los modelos necesarios para llegar al objetivo, e iré iterando sobre esta probando y testeando los resultados obtenidos sobre cada iteración para ir mejorando el resultado. Para al final generar el servició que contendrá el modelo a ejecutar.

1.4 Planificación del Trabajo

Este es el diagrama de Gantt temporal para el proyecto, sujeto a cambios durante la siguiente PAC donde la investigación de las actuales técnicas en la industria puede variar las tareas y la cantidad de tiempo de cada una.

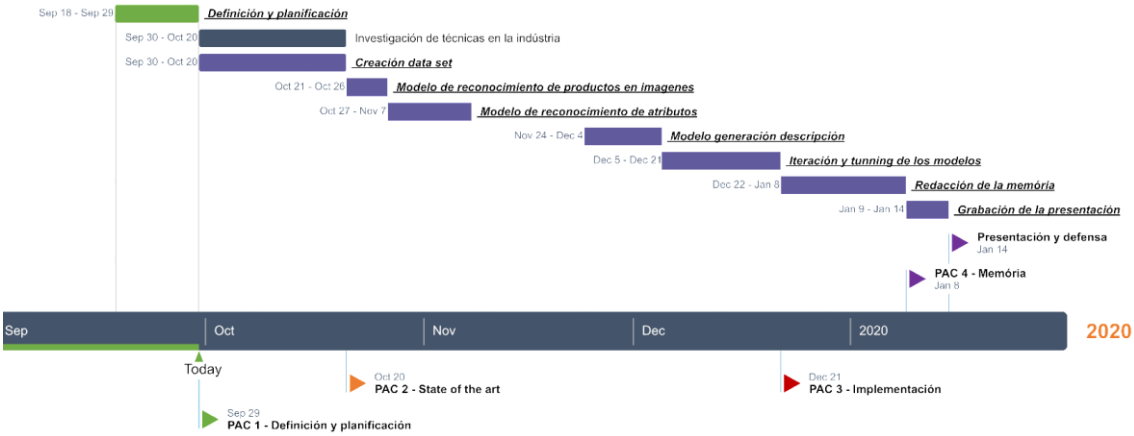

Figura 1. Diagrama Gantt del proyecto

*Nótese el hueco de dos semanas en noviembre debido a estar fuera del país.

1.5 Breve sumario de productos obtenidos

1. Modelo de reconocimiento de imágenes
2. Modelo para generar las descripciones
3. Servicio en la nube para ejecutar el modelo

1.6 Breve descripción de los otros capítulos de la memoria

- Estado del arte: Describe uno de los estándares en la industria para generar descripciones con lenguaje natural a partir de una imagen.
- Desarrollo del proyecto: Explica todos los pasos y etapas de la realización del proyecto y las pruebas que se han llevado a cabo para cada una.
- Conclusiones: Contiene la explicación del resultado del proyecto y como se podría seguir trabajando en él para mejorar.
- Repositorio del código y datos: Contiene los links para descargar el código del proyecto de github y los conjuntos de datos con los que se ha realizado.
- Ejecución en el cloud: Contiene una pequeña explicación de como crear el servicio para ejecutar el modelo en la nube de Amazon.

2. Estado del arte

Para entender el estado del arte en la generación automática de imágenes debemos primero entender que es Microsoft COCO (Common Objects in Context) [8], que consiste en un conjunto de imágenes con escenas complejas que contienen un conjunto de 91 tipos de objetos diferentes entre las 328 mil imágenes que lo forman. Para cada una de esas imágenes, crearon un pipeline capaz de anotar los diferentes tipos de objetos que hay presenten en cada imagen, acabando asó con unos 2.5 millones de etiquetas para todo el conjunto de imagen.

A partir de este conjunto de datos, en 2015 se creó un desafío para probar el mejor algoritmo para crear descripciones de imágenes utilizando este data set [3].

En esta competición, uno de los dos algoritmos que quedaron empatados en primer lugar fue desarrollado por Google, en el que combinaron una red neuronal convolucional (CNN), para detectar los diferentes objetos dentro de una imagen, y en vez de utilizarlo para dar una predicción con una capa final Softmax, eliminar la última capa para que sirva de input a una red neuronal recurrente (RNN) con arquitectura Long short-term memory (LSTM) (ver [9] [10]) que genere frases a partir de ese input para describir la imagen inicial. De esta forma, se pueden entrenar el modelo con un conjunto de datos que contenga tanto las imágenes como las descripciones para asegurarse que estas realmente tienen que ver con la imagen (ver [4]).

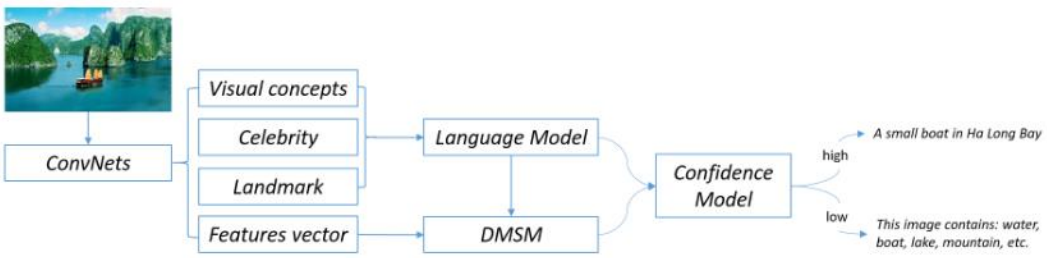

Figura 2. Pipeline de generación de imágenes de Microsoft

En 2016, Microsoft Research mejoró el pipeline anteriormente citada añadiendo varios modelos que eran capaces de detectar conceptos que agregan más información a la descripción bajo la premisa de que cierta información concreta es más importante que una descripción genérica [1]. Por eso, añadieron varias redes convolucionales para detectar famosos, paisajes concretos y conceptos visuales para pasárselo posteriormente a un modelo de lenguaje que juntamente con una red convolucional que genera un vector de `features` como en el modelo anterior de Google, sirven de input para un conjunto de Deep Multimodal Similarity Model (DMSM) [11] y Maximum Entropy Language Model (MELM) [12] ambos entrenados con el conjunto de datos de Microsoft COCO. Por último, el

pipeline termina con un modelo de confianza, que decide si crear una descripción muy concreta o más genérica.

Para probar la performance de este pipeline, se ha testeado con un conjunto de imágenes de MS COCO e imágenes aleatorias de Instagram, consiguiendo un incremento de 10% en descripciones excelentes y bajando en un 18% las descripciones malas. Por lo que este pipeline marca un nuevo estado del arte para la generación automática de imágenes especialmente para imágenes genéricas.

3. Desarrollo del proyecto

3.1. Creación del conjunto de datos

Para iniciar el proyecto, lo primero ha sido recopilar un conjunto de datos formado por imágenes de los productos a describir, para esto, he utilizado un script en Python para poder hacer `scraping` de la `cognitive API` de Microsoft la cual devuelve imágenes en función de un parámetro de búsqueda. De esta forma he podido descargar cientos de imágenes de diferentes categorías de productos. Una vez descargadas las imágenes, he obtenido algunas más de categorías en las que la API de Bing no ha devuelto muchos resultados y he limpiado duplicados o imágenes que realmente no tenían nada que ver con ninguno de los productos de manera manual.

Una vez obtenidas las imágenes, para este proyecto solo he creado las descripciones de las imágenes para las tarjetas de visita y los `banners`, ya que he considerado que dos productos diferentes era suficiente para probar si el modelo podía funcionar correctamente o no. Por lo que el conjunto de datos final, son carpetas con imágenes de tarjetas de visita y `banners` en conjunto con un csv con una columna que contiene el nombre de la imagen y su descripción.

Como dato adicional, dentro de la carpeta ‘Scripts’ del código del proyecto, además del script para obtener las imágenes de la `cognitive API` de Bing, hay otro script para transformar las imágenes en formato webpg a jpg, ya que como parte del `scraping`, algunas de las imágenes obtenidas tenían este formato de imagen pensado para la web, que no funciona correctamente como muchas de las librerías de Python. De esta forma este script es parte de la ‘limpieza’ de los datos.

3.2. Desarrollo del modelo predictivo

Para comenzar a desarrollar el proyecto, lo primero que hice fue hacer un modelo de clasificación de productos, simplemente para verificar que los datos obtenidos eran los suficientes y tenían la calidad suficiente como para que el modelo fuera capaz de encontrar patrones diferenciales en las imágenes de diferentes tipos de productos.

Para ello, lo primero que hice fue crear una red neuronal convolucional con una capa softmax al final que entrenara con unos 12 productos diferentes para ver si la clasificación funcionaba correctamente.

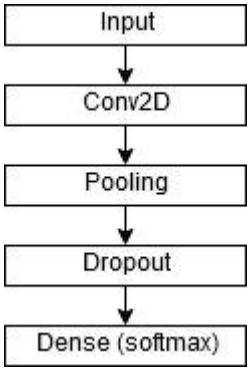

Figura 3. Primera iteración de la CNN

Naturalmente, con una CNN tan simple los resultados no fueron buenos, consiguiendo simplemente un `accuracy` de 20%. Así que continué ampliando el conjunto de datos y iterando sobre la CNN añadiendo nuevas capas hasta conseguir un `accuracy` de alrededor de 40%. Hasta que llegué al punto en el que no tenia forma de conseguir realmente muchas más imágenes de calidad de este tipo de productos, así que recurrí a utilizar `data augmentation` [15] que consiste en realizar transformaciones en las imágenes para generar nuevas.

Por ejemplo, de la imagen original, podemos crear una nueva aplicando un desplazamiento de la imagen hacia un lado, modificando el contraste para cambiar los colores, rotando la imagen, realizando un escalado, etc. De esta forma multiplicamos nuestro conjunto de datos de manera artificial.

Con estos nuevos datos, conseguí llegar cerca de un 57% de `accuracy`, pero seguía siendo insuficiente, así que el siguiente proceso al que recurrí fue el `transfer learning` [14], que consiste en reutilizar un modelo ya desarrollado y entrado como punto de partida para uno nuevo, pudiendo así reutilizar lo que ese modelo había aprendido en el nuestro.

Así que utilicé la arquitectura VGG16 [13] como punto de partida del modelo, pasando así las `features` obtenidas de este hacia un modelo con dos capas Relu y una última softmax para realizar la predicción de la clasificación.

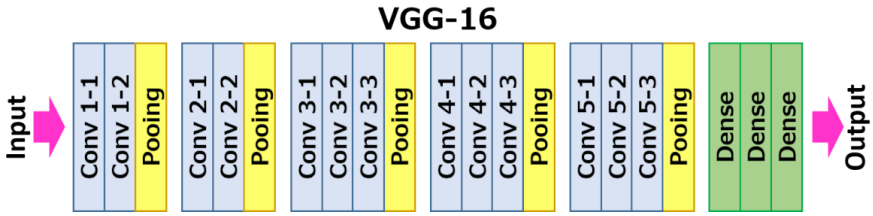

Figura 4. Arquitectura VGG16

Con este método, el `accuracy` del modelo llegó a un 75%, por lo que conseguí una mejora significativa de este, pero todavía insuficiente. Llegados a este punto, ya no podía conseguir más imágenes de manera fácil, y no veía como mejorar el modelo, así que con prueba y error, empecé a eliminar transformaciones de la

generación de datos artificiales con el `data augmentation` de manera que acabé dejando solamente un escalado de las imágenes, potenciando así el `accuracy` hasta un 87% que ya consideré aceptable dada la dificultad de conseguir imágenes.

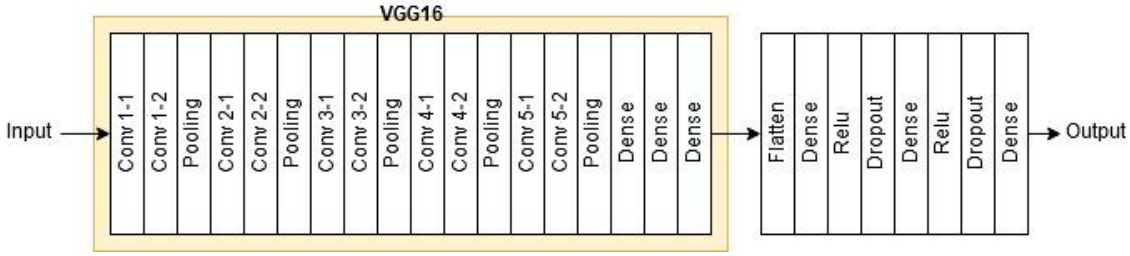

Figura 5. Modelo de clasificación final

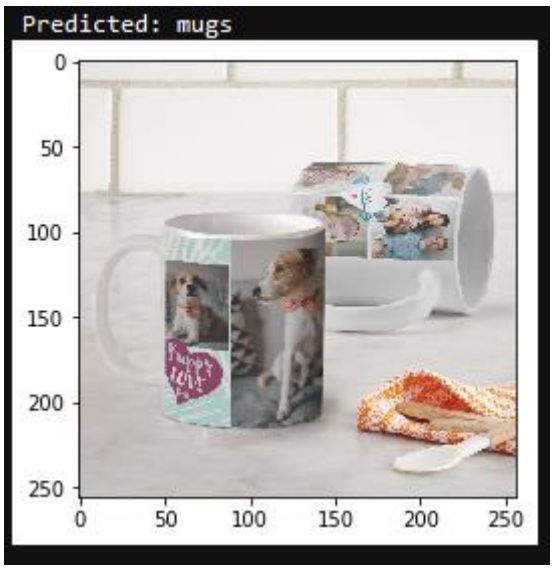

Figura 6. Test del modelo de clasificación en una imagen de una taza

Una vez cómodo con el conjunto de datos y con diversas técnicas para mejorar los modelos y detectar correctamente los objetos en ellas, trabajé en el modelo para predecir las descripciones de estas.

Para ello la única parte a cambiar es la parte posterior al VGG16, por lo que opté por conectarlo a un modelo que aceptara dos inputs, el resultado del VGG16 i una parte de la descripción. De manera que se iterara sobre esa descripción parcial intentado predecir la siguiente palabra hasta que se llega a un token final.

Para conseguir esto, hay que modificar el conjunto de datos para modificar las descripciones y añadirle un token de inicio y de fin a cada una (en mi caso #START y #END). De esta forma, pasamos la imagen por la primera parte del modelo, obteniendo así un vector con sus `features` y le pasamos al modelo el primer token de inicio, repitiendo el proceso hasta que el modelo predice el token final.

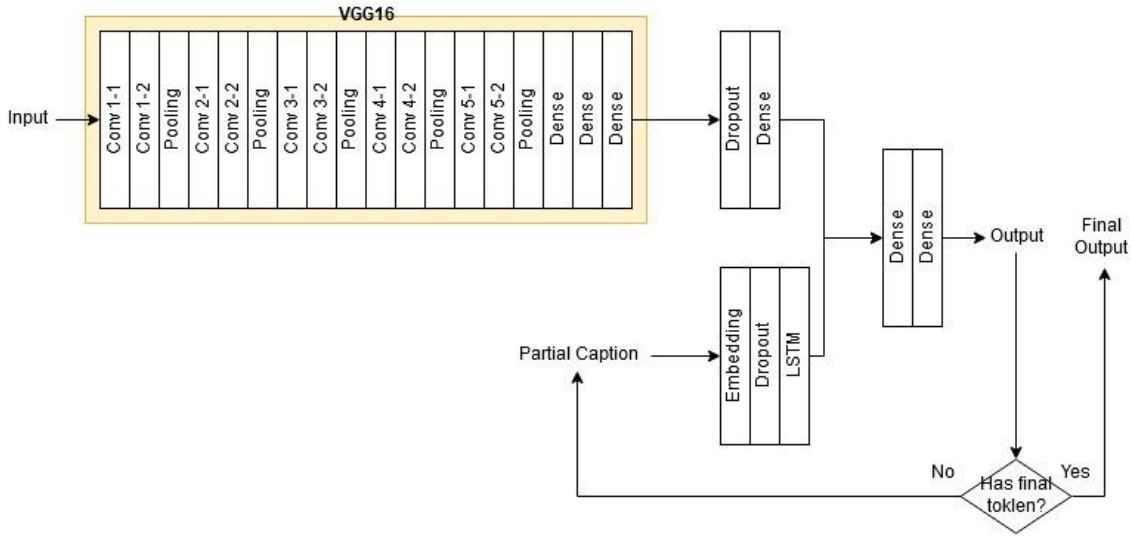

Figura 7. Diagrama del modelo de predicción de descripciones en primera fase

Al probar de manera manual los resultados de este modelo, estaba obteniendo muchas veces descripciones inacabadas, o con palabras repetidas continuamente como por ejemplo: `Two banners from the back back back back back`. De manera que el resultado no era positivo en la gran mayoría de casos.

Después de investigar e iterar sobre el modelo para mejorarlo, acabe utilizando GloVe [16] para utilizarlo en la capa de `Embedding` y utilizar los pesos y no reentrenarlo con el corpus limitado producido por mi conjunto de datos, consiguiendo un rendimiento mucho mejor en pruebas manuales.

Por último, acabe modificando la primera capa del VGG16 para utilizar el modelo Inception V3 de Google [17] [18], que me acabó dando mejores resultados en las descripciones finales.

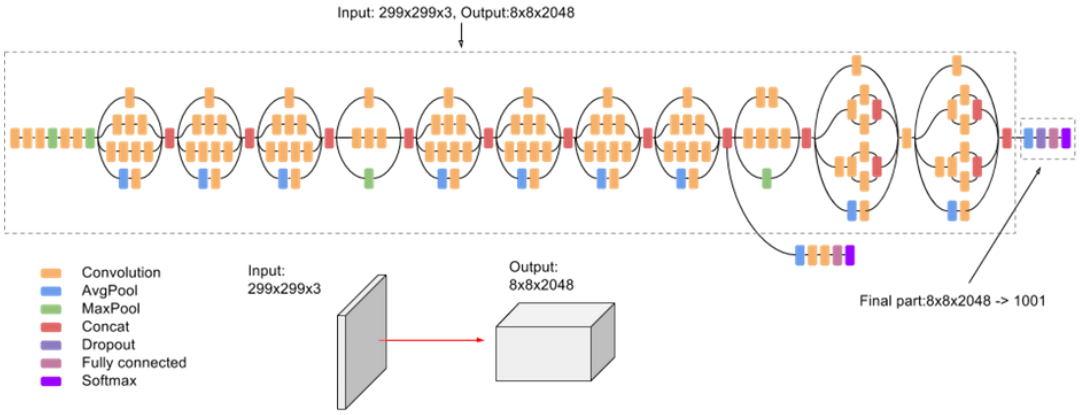

Figura 8. Arquitectura del modelo Inception V3 de Google

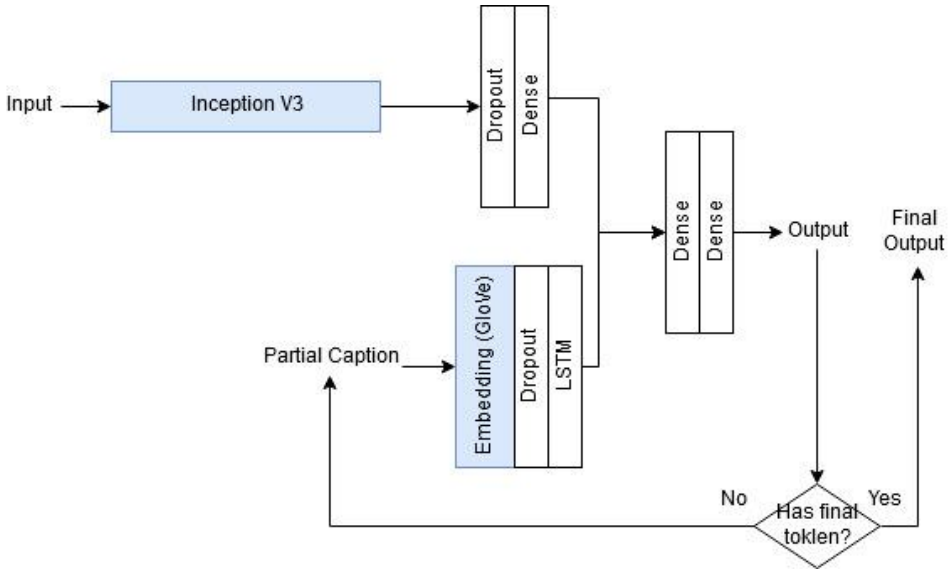

Figura 9. Modelo de preddicion de descripciones con Inception v3 y GloVe

Con estos cambios, las descripciones obtenidas eran mucho más similares a las originales y acertadas, pero había errores en las características de los productos, como, por ejemplo, confundir las tarjetas de visita de cantos cuadrados con las redondeadas, lo cual era de esperar dado el limitado número de imágenes disponibles y que la cantidad de estas con cantos redondeados era inferior a las otras.

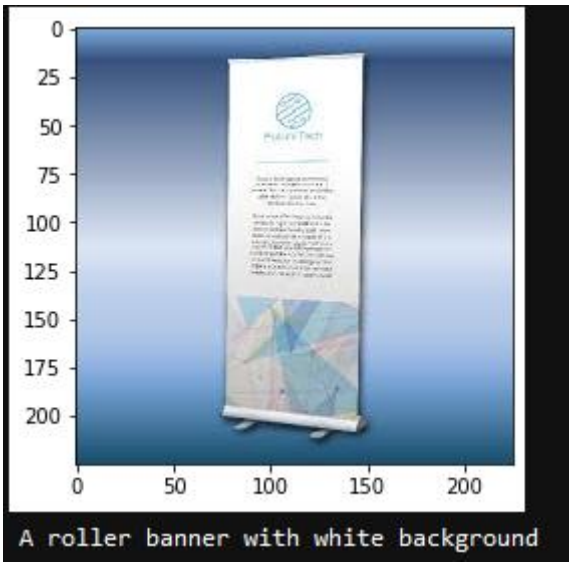

Figura 10. Predicción de la descripción de la imagen correcta

Por lo que primero que pensé, fue en utilizar el modelo de clasificación inicial para clasificar las imágenes según sus atributos, obteniendo así un conjunto de atributos para pasarle a la capa LSTM y modificar su `Bias` en función de estos atributos para que le diera más peso a las palabras que concordaran con estos como se propone en el `paper` [19].

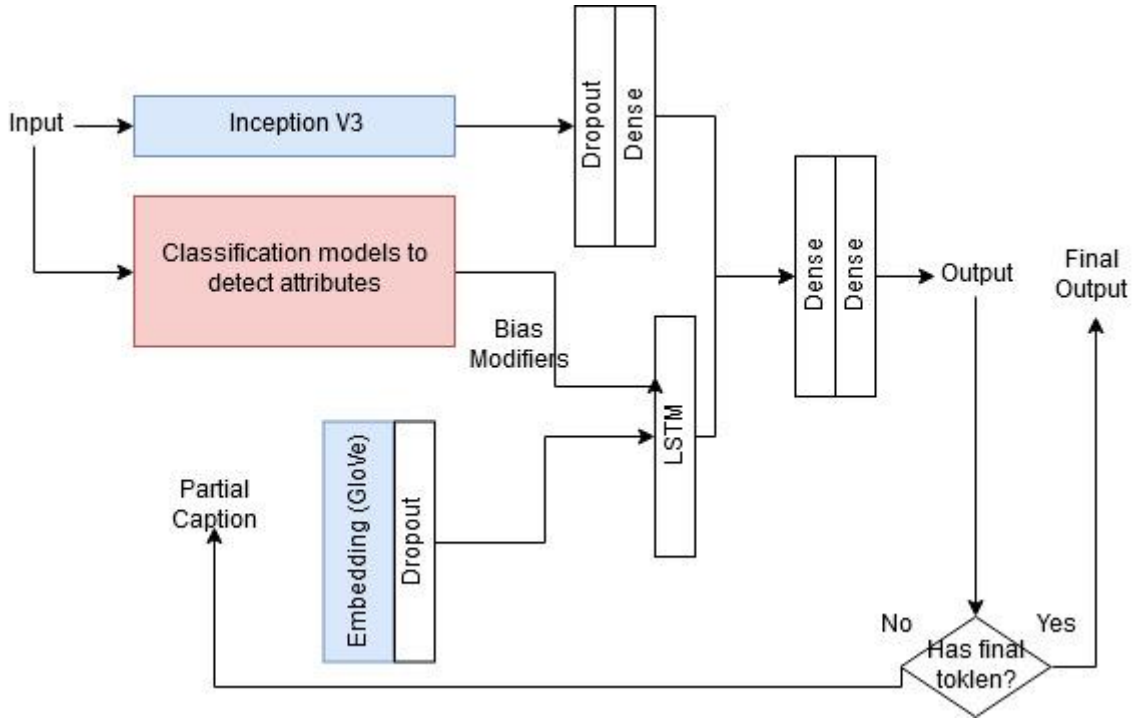

Figura 11. Modelo conceptual de predicción de texto alterada por atributos

Por desgracia las clases utilizadas en la librería de Keras no están preparadas para este caso, y mis intentos de implementar-lo por mí mismo basándome en la clase original de Keras no acabaron de funcionar correctamente, así que por falta de tiempo acabé descartando la idea de implementarlo como parte del modelo. Así que acabé creando un pipeline de modelos para intentar solventar este problema. De manera que el pipeline utiliza el primer modelo de clasificación de productos que cree, de esta manera según el producto que detecte, la imagen se pasa a otros modelos para detectar sus atributos, en este caso si es una `banner` se detecta de que tipo es (vinilo, tela o `Roll-ups`) mientras que para las tarjetas de visita detecta los cantos. Así obtenemos modelos especializados en pequeñas tareas que darán un mejor rendimiento que un modelo genérico dada la muy limitada cantidad de datos de los que se dispone para este tipo de productos, después entonces predecimos la descripción con el modelo anterior y comprobamos que en la descripción las características son correctas.

Sin duda, este sistema solamente puede ser viable en casos como este, donde tienes una cantidad pequeña de productos diferentes con pocos atributos a detectar en cada uno, para casos más genéricos seria imposible construir tantos modelos diferentes.

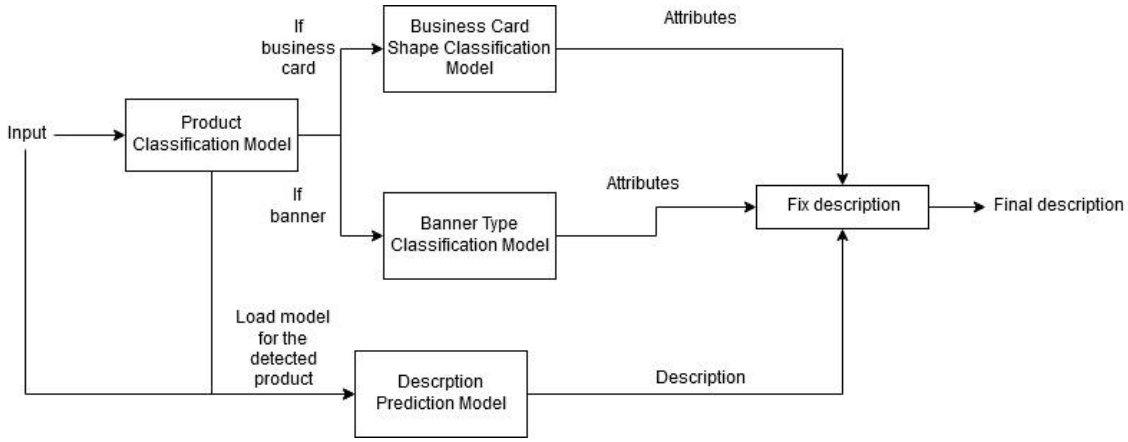

Figura 12. Pipeline final para la generación de descripciones

4. Conclusiones

Al terminar el proyecto, queda claro que la cantidad de datos necesaria para este tipo de modelos es mucho mayor de la que se disponía, a eso se debe el no poder llegar ni a un 90% de `accuracy` en ninguno de los modelos y el necesitar crear un `pipeline` y modelos específicos para poder obtener un buen resultado, los cuales aún así son muy susceptibles a encadenar errores dado el margen de error de cada uno de ellos. Aunque con mejores datos, se podría conseguir un rendimiento mejor para cada uno de los modelos, y teniendo en cuenta que la cantidad de productos en este caso es finita y no muy alta, es viable el crear múltiples modelos para cada uno de los diferentes productos.

Aún así, la cantidad de datos no lo es todo, como se ha podido comprobar al utilizar `data augmentation`, simplemente el generar datos artificiales modificando las imágenes de las que se disponían acabaron generando ruido en el conjunto de datos haciendo que el modelo lograra peores resultados. Por lo que queda claro que los datos han de ser de calidad para que influyan positivamente en el rendimiento del modelo.

Pese a haber conseguido generar descripciones para las imágenes, estás han acabado siendo simples y no usables en un entorno de producción, es necesario obtener muchas más imágenes como datos para poder detectar todos los atributos de forma correcta para cada uno de los productos. Además, el proceso de crear las descripciones de cada una de las imágenes a mano es lento y tedioso, por lo que sería difícil conseguir un conjunto de datos muy amplio y poder generar las descripciones. Por este motivo al final el proyecto solo dispone de modelos para tarjetas de visitas y `banners`.

En cuanto a la metodología seguida, ha sido la planteada inicialmente en la planificación del proyecto, comenzando por obtener las diferentes imágenes de cada uno de los productos, la creación del primer modelo de clasificación del producto, seguido por el modelo de clasificación de los atributos de cada producto y finalmente el de generación de descripciones. La única diferencia con el diagrama de Gantt es que el proceso de iteración y `tunning` de los modelos se ha realizado en cada uno de los modelos antes de pasar al siguiente.

Como futuras líneas de trabajo, la recopilación de un mejor conjunto de datos es la base para mejorar este proyecto, el problema es que no hay una gran cantidad de datos disponibles para productos tan específicos para el nivel de granularidad que se pretende llegar en los atributos de cada uno. Una vez conseguido esto, mejorar el modelo LSTM para que reciba como parámetro los atributos del producto además de la descripción parcial para que estos influyan en el `bias` de la predicción y así mejorar las opciones de que la descripción final contenga las características del producto correctas.

Estas serian las dos formas de mejorar el proyecto en gran medida a partir de este punto.

5. Glosario

- COCO: Common Objects in Context
- CNN: Convolutional Neural Network
- RNN: Recurrent Neural Network
- DMSM: Deep Multimodal Similarity Model
- MELM: Maximum Entropy Language Model
- LSTM: Long short-term memory
- AWS: Amazon Web Services
- CDK: Cloud Development Kit

6. Bibliografía

[1] Kenneth Tran, Xiaodong He, Lei Zhang, Jian Sun, Cornelia Carapcea, Chris Thrasher, Chris Buehler y Chris Sienkiewicz. Rich Image Captioning in the Wild, 2016.

[2] Oriol Vinyals, Alexander Toshev, Samy Bengio y Dimitru Erhan. Show and Tell: A Neural Image Caption Generator, 2014.

[3] https://ai.googleblog.com/2016/09/show-and-tell-image-captioning-open.html, visitada el 8 de octubre de 2019.

[4] https://ai.googleblog.com/2014/11/a-picture-is-worth-thousand-coherent.html, visitada el 8 de octubre de 2019.

[5] Bernardi, R., Cakici, R., Elliott, D., Erdem, A., Erdem, E., Ikizler-Cinbis, N., ... Plank, B. (2016). Automatic Description Generation from Images: A Survey of Models, Datasets, and Evaluation Measures. Journal of artificial intelligence research, 55, 409-442. [4900]. https://doi.org/10.1613/jair.4900.

[6] https://machinelearningmastery.com/how-to-caption-photos-with-deep-learning/, visitada el 10 de octubre de 2019.

[7] https://blog.goodaudience.com/automatic-image-captioning-building-an-image-caption-generator-from-scratch-4bdd8744bc38, visitada el 12 de octubre de 2019.

[8] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick y Piotr Dollár. Microsoft COCO: Common Objects in Context, 2015.

[9] Sepp Hochreiter y Jürgen Schmidhuber. Long Short-term memory, 1997

[10] Alex Sherstinsky. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network, 2018.

[11] Jacob Devlin, Hao Cheng, Hao Fang, Saurabh Gupta, Li Deng, Xiaodong Hew, Geoffrey Zweig y Margaret Mitchell. Language Models for Image Captioning: The Quirks and What Works, 2015.

[12] Adam L. Berger, Stephen A. Della Pietra y Vincent J. Della Pietra. A Maximum Entropy Approach to Natural Language Processing, 1996.

[13] Karen Simonyan y Andrew Zisserman. Very Deep Convolutional Networks For Large-scale Image Recognition, 2015

[14] Beenish Zia, Ramesh Illikkal, Bob Rogers. Use Transfer Learning For Efficient Deep Learning Training On Intel Xeon Processors, 2018

[15] Mikołajczyk, Agnieszka y Grochowski, Michał. (2018). Data augmentation for improving deep learning in image classification problem. 117-122. 10.1109/IIPHDW.2018.8388338.

[16] Jeffrey Pennington, Richard Socher, Christopher D. Manning. GloVe: Gloval Vectors for Word Reperesentation, 2014

[17] Jyotsna Bankar y Nitin R Gavai. Convolutional Neural Network based Inception v3 Model for Animal Classification, 2018.

[18] https://codelabs.developers.google.com/codelabs/cpb102-txf-learning/index.html#0, visitada el 3 de diciembre de 2019.

[19] Ting Yao, Yingwei Pan, Yehao Li, Zhaofan Qiu y Tao Mei. Boosting Image Captioning with Attributes, 2016.

7. Anexos

6.1. Repositorio del código y datos

El código del proyecto se puede encontrar en el repositorio de Github https://github.com/xmedinat/TFM, en el fichero Readme.md están las instrucciones para poder ejecutar el código.

Los datos y modelos utilizados se pueden obtener en:

- Datos: https://drive.google.com/file/d/1vdtQLlP78jW4hGh-EMLOIvrQJoSyEs4X/view?usp=sharing
- Modelos: https://drive.google.com/file/d/1iGk1ar9lgJXi1otIEh-5aZWHLcBgYRb-/view?usp=sharing

6.2. Ejecución en el cloud

Dentro de la carpeta `Infrastructure` del proyecto, se pueden encontrar los scripts de AWS CDK para crear la infraestructura necesaria para crear una función en AWS Lambda (`Serverless`) que ejecute la versión del modelo de predicción de texto sin la corrección posterior (ya que por limitaciones de espacio del código en AWS Lambda la versión completa no podía ejecutarse).

Para hacerlo funcionar, es necesario un usuario de AWS con permisos para todos los servicios asociados como esta detallado en el Readme.md del proyecto, allí se encuentran también los comandos para hacerlo funcionar.

Una vez completado el `stack` de AWS, dentro de API Gateway se encontrará la URL final a la que llamar para utilizar el servicio, que espera un json en el `body` de la petición con una imagen en base64 en un campo llamado `image`.