Colección Revistas Social

Javier Lima García Álvaro Luis González Brito
Marlon Díaz Pérez Álbaro Suárez Valdés
David Lezcano Becerra Alfredo Nuño Oquendo

February 24, 2025

1 Introducción

1.1 Contexto y motivación

Todos los días, nos encontramos con una gran cantidad de imágenes de diversas fuentes como redes sociales, medios digitales, documentación técnica, entre otros, las cuales los espectadores muchas veces tienen que interpretar ellos mismos ya que no tienen una descripción, pero el ser humano puede entenderlas sin necesidad de subtítulos detallados. Sin embargo, con el aumento de los volúmenes de datos ha surgido la necesidad de tenerlos bien descritos para ser almacenados y poder buscarlos de manera eficiente. Dado que la descripción manual de imágenes es un proceso costoso y poco escalable, la automatización de esta tarea se ha vuelto una prioridad en el desarrollo de tecnologías inteligentes.

1.2 Problema a resolver

La Oficina del Historiador de La Habana dispone de un acervo de más de 200 ejemplares pertenecientes a la Colección de Revistas Social, los cuales contienen un volumen significativo de imágenes que requiere ser almacenadas junto con su respectiva descripción para facilitar su catalogación y recuperación. No obstante, una proporción considerable de estas imágenes carece de descripciones explícitas dentro de las publicaciones. A pesar de ello, es

posible inferir información relevante a partir del contexto proporcionado en la revista o mediante el análisis del contenido visual de las imágenes.

1.3 Objetivo del proyecto

El proyecto busca generar descripciones automáticas para las imágenes de la Colección de Revistas Social de la Oficina del Historiador de La Habana, donde muchas carecen de textos explicativos. Para ello, se emplearán los modelos preentrenados *CLIP* y *BLIP*, que combinan visión por computadora y procesamiento de lenguaje natural para interpretar el contenido visual y generar descripciones precisas.

2 Marco Teórico

2.1 Conceptos claves

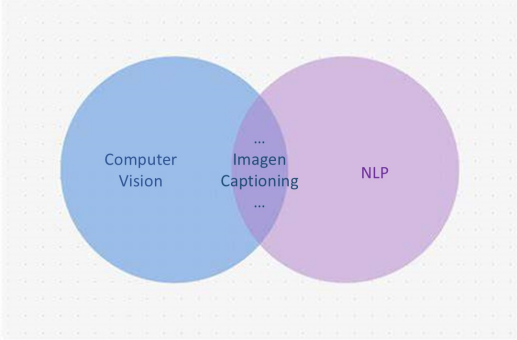

Figure 1: Imagen representativa de la técnica de Image Captioning

Image Captioning es el proceso de generar descripciones en lenguaje natural para imágenes. Combina técnicas de visión por computadora y procesamiento del lenguaje natural.

En la última década, ha habido avances significativos en esta área, con modelos evolucionando desde enfoques basados en redes neuronales recurrentes (*RNN*) hasta arquitecturas avanzadas de transformers y modelos multimodales. Los primeros enfoques exitosos en *Image Captioning* utilizaron una combinación de redes neuronales convolucionales (*CNN*) para extraer características visuales y redes neuronales recurrentes (*RNN*), como *Long*

Short-Term Memory (*LSTM*), para generar texto secuencialmente. Uno de los modelos más influyentes en esta categoría es *Show and Tell* (Vinyals et al., 2015), el cual emplea una *CNN* preentrenada (*Inception-v3*) para obtener representaciones de imágenes y luego usa un *LSTM* para generar descripciones de manera secuencial. Aunque efectivo, este modelo tiene limitaciones al generar descripciones demasiado genéricas o poco precisas debido a la falta de un mecanismo de atención.

Los avances en modelos de lenguaje, como *Transformer* (Vaswani et al., 2017), llevaron a una nueva generación de modelos de *Image Captioning* que reemplazaron las *RNN* con mecanismos de auto-atención más eficientes. Uno de los primeros enfoques en adoptar transformers fue *Image Transformer* (Parmar et al., 2018), que aplicó *self-attention* directamente sobre las imágenes en lugar de utilizar *CNNs*. Sin embargo, este modelo tenía problemas de escalabilidad.

En los últimos años, los modelos multimodales han revolucionado el campo del *Image Captioning*, combinando redes neuronales de visión y modelos de lenguaje avanzados para mejorar la calidad de las descripciones.

2.2 Modelos utilizados

CLIP (*Contrastive Language-Image Pretraining*), desarrollado por OpenAI (Radford et al., 2021), es un modelo multimodal que aprende a relacionar imágenes y texto a través de un entrenamiento contrastivo. Su principal ventaja es que permite interpretar imágenes en función de descripciones en lenguaje natural sin necesidad de un entrenamiento supervisado específico para cada tarea.

CLIP tiene dos componentes principales:

- **Encabezado de Imagen:** Un Vision Transformer (ViT) o una CNN (como ResNet) para extraer representaciones visuales.
- **Encabezado de Texto:** Un modelo de lenguaje tipo Transformer, similar a BERT, para convertir frases en representaciones numéricas.

Ambos módulos transforman imágenes y textos en un espacio de características compartido donde se pueden comparar directamente.

El entrenamiento contrastivo de **CLIP** se basa en la idea de maximizar la similitud entre pares correctos de imágenes y textos mientras minimiza la

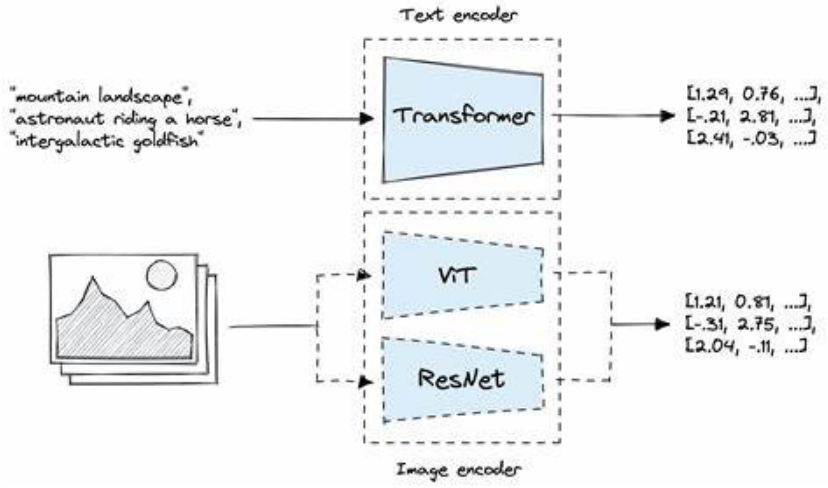

Figure 2: Arquitectura de CLIP.

similitud entre pares incorrectos. Para lograr esto, **CLIP** se entrena con un gran conjunto de datos de imágenes y sus descripciones de internet. Cada imagen y su texto correspondiente se convierten en representaciones numéricas mediante las redes neuronales vistas anteriormente. Luego, el modelo calcula la similitud entre todos los pares en un mismo lote mediante un producto escalar en un espacio de características compartido. Se emplea una función de pérdida contrastiva (*InfoNCE Loss*), la cual aumenta la proximidad entre los vectores de imágenes y textos correctos y aleja los no relacionados. A través de este proceso, **CLIP** aprende una representación multimodal donde imágenes y descripciones semánticamente similares quedan cercanas en el espacio, permitiéndole realizar tareas de clasificación, búsqueda de imágenes y razonamiento visual-lingüístico sin necesidad de reentrenamiento específico (*zero-shot learning*).

A pesar de esto, **CLIP** no puede generar captions directamente, sino en combinación con modelos generativos como **BLIP** (*Bootstrapped Language-Image Pretraining*) (Li et al., 2022), que combina *Vision Transformers* (*ViT*) con modelos de lenguaje como *BERT* para generar captions más precisos y ricos en contexto. **BLIP** es capaz de realizar tanto *Image Captioning* como tareas relacionadas, como *Visual Question Answering* (*VQA*) y *Text-Based Image Retrieval*. Gracias a su enfoque de entrenamiento basado en múltiples objetivos, **BLIP** supera a muchos modelos previos en métricas como *BLEU*

y *CIDEr*.

Introduce una arquitectura llamada *Multimodal Mixture of Encoder-Decoder (MED)*, que permite un preentrenamiento más flexible en comparación con modelos previos como **CLIP**, combinando tres enfoques:

- Codificador unimodal para el alineamiento de imágenes y textos.
- Codificador basado en imágenes que incorpora atención cruzada para capturar relaciones más detalladas.
- Decodificador basado en imágenes que genera texto condicionalmente.

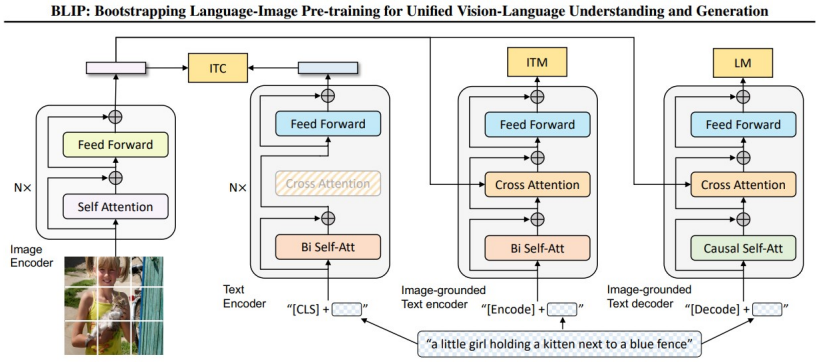

Figure 3: Arquitectura de BLIP.

Esta combinación es lo que permite que **BLIP** se desempeñe bien en tareas tanto de comprensión como de generación, mientras que **CLIP** no. En términos de entrenamiento, **BLIP** introduce el método *CapFilt (Captioning + Filtering)* para mejorar la calidad de los datos utilizados en el preentrenamiento. A diferencia de **CLIP**, que entrena en grandes cantidades de pares imagen-texto sin filtrar, **BLIP** utiliza un generador de subtítulos (*Captioner*) para crear descripciones sintéticas y un filtro (*Filter*) para eliminar textos ruidosos, asegurando que el modelo aprenda de datos más limpios y relevantes.

YOLO (*You Only Look Once*) es un algoritmo de detección de objetos que destaca por su velocidad y eficiencia.

YOLO divide la imagen en una rejilla, donde cada celda es responsable de predecir objetos cuyo centro caiga dentro de ella. Para cada celda, se predicen los siguientes elementos:

- **Cajas delimitadoras (Bounding Boxes)**: Múltiples cajas de diferentes tamaños y formas.
- **Confianza (Confidence Score)**: La probabilidad de que la caja contenga un objeto y qué tan precisa es la caja delimitadora.
- **Clases de objetos**: La probabilidad de que el objeto dentro de la caja pertenezca a cada una de las clases que el modelo conoce.

Tras la fase de predicción, se aplican dos técnicas para refinar los resultados:

- **Umbral de confianza**: Se eliminan las cajas con una confianza baja.
- **Supresión no máxima (Non-Maximum Suppression - NMS)**: Si varias cajas se superponen y predicen el mismo objeto, NMS elige la caja con la mayor confianza y elimina las demás.

El resultado final son las cajas delimitadoras que rodean los objetos detectados en la imagen, junto con sus respectivas clases y niveles de confianza.

2.3 Trabajos relacionados

ClipCap es un modelo que utiliza las codificaciones de **CLIP** como prefijo para las descripciones textuales. Emplea una red de mapeo simple sobre la codificación obtenida y luego ajusta un modelo de lenguaje para generar descripciones coherentes de las imágenes. Este enfoque ha demostrado ser eficiente y logra resultados comparables al estado del arte en conjuntos de datos como *nocaps* (Mokady, Bites, Sadeh, 2021).

Fine-grained Image Captioning with CLIP Reward propone el uso de **CLIP** como una función de recompensa para mejorar la precisión de las descripciones generadas. Al calcular la similitud multimodal, se guía al modelo de generación para producir descripciones más detalladas y precisas (Zhan Wu, 2022).

BLIP-2 es una versión avanzada de **BLIP** que puede responder preguntas sobre imágenes y generar subtítulos. **BLIP-2** utiliza una estrategia de preentrenamiento eficiente que combina modelos de visión preentrenados y modelos de lenguaje extenso, superando a modelos anteriores en tareas como **VQAv2** y estableciendo un nuevo estado del arte en subtítulos de *zero-shot* (Li, Lu, Wu, 2023) .

3 Metodología

3.1 Arquitectura

La arquitectura del sistema está compuesta por los siguientes componentes:

- **Preprocesamiento de datos**
- **Entrenamiento**
- **Combinación**
- **Evaluación**

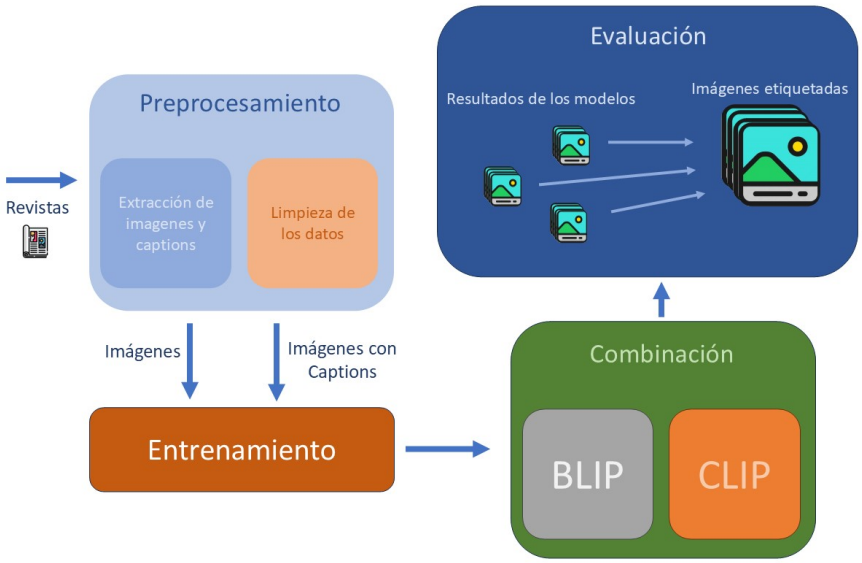

Figure 4: Arquitectura del sistema.

3.2 Preprocesamiento de Datos

Entre los recursos para el proyecto contamos con:

- 239 ejemplares de la Colección de Revistas Social.

- Índice analítico de la revista Social, 1916-1938.

El problema que abordamos consiste en trabajar con un conjunto de revistas antiguas que han sido previamente escaneadas. Nuestro objetivo es extraer las imágenes contenidas en estas revistas junto con sus descripciones originales, si las tienen. En los casos en que no exista una descripción, el sistema deberá generarla automáticamente.

3.2.1 Extracción de Imágenes de las Revistas

La extracción de imágenes no era viable mediante métodos convencionales, ya que cada página de las revistas escaneadas se almacenaba como una única imagen. Por ello, se requirió un modelo de detección que permitiera identificar y extraer las fotografías de cada página. Tras analizar el estado del arte, seleccionamos **YOLO** por su eficiencia y precisión en la detección de objetos, permitiendo no solo identificar imágenes, sino también sus posibles descripciones [**Redmon2016**].

Para adaptar **YOLO** a la detección de imágenes en las revistas, construimos una base de datos personalizada. Seleccionamos 14 de las 239 revistas, las dividimos por páginas y las subimos a *Roboflow* para el etiquetado manual. Definimos dos clases: **imágenes** y **captions**. Dado que los *captions* no tenían un formato fijo, los etiquetamos junto con la región de la imagen que contenía el texto, permitiendo diferenciarlos del resto del contenido de la revista.

Esas 14 revistas nos permitieron crear una base de datos de 770 fotos. Dado que este tamaño no es suficiente para entrenar redes neuronales como **YOLO**, que requieren bases de datos de miles de imágenes, decidimos aplicar técnicas de *data augmentation* como:

- Conversión a escala de grises, para hacer que el modelo sea independiente del color.
- *Mixup*, que permite crear nuevas imágenes combinando imágenes aleatorias y sus respectivas etiquetas.

Una vez aplicadas estas técnicas, obtuvimos una base de datos de alrededor de 1000 imágenes.

De las 239 revistas, finalmente obtuvimos más de 25,000 imágenes.

3.2.2 Extracción de Captions de las Revistas y Texto Relacionado

Tras extraer las imágenes y *captions*, utilizamos *Tesseract*, una herramienta de *OCR*, para obtener el texto de las imágenes etiquetadas como *captions*. En general, el desempeño fue bueno, aunque en algunos casos presentó errores en el reconocimiento de caracteres. Al analizar otras opciones más precisas, encontramos que la mayoría eran de pago, por lo que *Tesseract* se convirtió en la mejor alternativa accesible para nuestro proyecto.

Para cada imagen, se guardó no solo el *caption* asociado en la revista, sino también el texto circundante. Esto se hizo porque, en los casos donde no había un *caption*, el texto alrededor podría ayudar al modelo a generar una descripción apropiada.

3.2.3 Limpieza de los Datos

Como se mencionó anteriormente, en muchos casos el *caption* extraído de las imágenes no era coherente o no describía correctamente la imagen. Para entrenar el modelo que generaría los *captions* faltantes, necesitábamos imágenes con sus *captions* lo más precisos posibles. Por lo tanto, era necesario realizar una limpieza de los *captions* que *Tesseract* extraía incorrectamente o que no estaban bien relacionados con sus imágenes asociadas.

Para esto, realizamos un filtrado en 3 etapas:

- Filtrado de ruido
- Retocado de captions
- Solidez de los datos

En el filtrado de ruido analizamos para cada caption en la base de datos qué porcentaje de caracteres inválidos presenta, y si es superior al 70% eliminamos el caption.

A los captions resultantes de la primera etapa se les aplica un autocorrector que elimina saltos de linea y el resto de caracteres inválidos, para luego utilizando un *LLM* (específicamente Mistral) darle sentido a los captions.

Por último, apoyándonos en el modelo **CLIP**, proporcionando como entrada tanto la imagen como su *caption* filtramos aquellos captions que no describen correctamente la imagen. Sin embargo, **CLIP** no podía evaluar directamente la calidad de un *caption*. Por eso, generamos 5 *captions* alternativos utilizando oraciones de un artículo para cada imagen y se lo damos

como entrada al modelo, si alguno de los *captions* generados obtenía una mayor probabilidad de estar relacionado con la imagen, asumíamos que el *caption* original no era válido y lo eliminamos.

3.3 Implementación del modelo

Para abordar el problema de agregar etiquetas a las fotos de las revistas, nos apoyamos en los modelos preentrenados **CLIP** y **BLIP**, los cuales muestran un buen rendimiento según el estado del arte y son de código abierto.

Sin embargo, debido a las particularidades de nuestro proyecto, decidimos combinarlos y ajustarlos a los datos específicos de este. La combinación se realizó de la siguiente manera:

Para **CLIP**, utilizamos como *captions* fragmentos del texto alrededor de las imágenes. **BLIP**, por otro lado, solo requiere las imágenes para generar los *captions*.

En primer lugar, aplicamos los modelos **CLIP** y **BLIP** de la forma estándar: **BLIP** genera un *caption* para cada imagen, mientras que para **CLIP** seleccionamos el *caption* con la mayor probabilidad asociada a cada imagen. A continuación, utilizamos algunas imágenes con *captions* obtenidos manualmente para realizar un proceso de *fine-tuning* de ambos modelos. Con estos modelos ajustados, intentamos nuevamente generar *captions* para las imágenes.

Además, implementamos un enfoque híbrido con dos estrategias diferentes:

- Usamos **BLIP** para generar un *caption* para la imagen y lo combinamos con los *captions* del texto alrededor de la imagen. Este conjunto se utiliza como entrada para **CLIP**, y el resultado con la mayor probabilidad es el que se devuelve como la descripción final de la imagen.

- Usamos **BLIP** para generar un *caption* para la imagen y, luego, aplicamos **CLIP** de la manera descrita previamente solo con el texto circundante de la imagen. Si el mejor resultado obtenido de **CLIP** tiene una probabilidad suficientemente alta, lo usamos como *caption*. En caso contrario, nos quedamos con el *caption* generado por **BLIP**.

3.4 Estrategia de entrenamiento y evaluación

4 Estrategia de entrenamiento y evaluación

Para el entrenamiento del modelo, se utilizó un conjunto de 1700 imágenes con sus respectivos *captions*, obtenidas tras el proceso de limpieza de datos. Inicialmente, se consideró la implementación de un esquema de validación cruzada *k-fold* con $k = 10$ para evaluar el impacto de los ajustes en los hiperparámetros del modelo. Sin embargo, debido a las limitaciones computacionales impuestas por la infraestructura de Google Colab, fue necesario adoptar una estrategia alternativa.

En su lugar, se optó por dividir aleatoriamente los datos en 10 conjuntos, cada uno compuesto por 250 imágenes para entrenamiento y 50 imágenes para validación. Este enfoque permitió realizar la evaluación del modelo de manera eficiente, asegurando un balance entre la exploración de distintos ajustes en los hiperparámetros y la viabilidad computacional del experimento.

Para establecer la superioridad de un modelo sobre otro de manera estadísticamente consistente, utilizamos métricas de similitud de texto, como **METEOR**, **BLEU** y **ROUGE**, que son comunes en tareas de generación de lenguaje natural (NLG).

5 Análisis de experimentos y resultados

5.1 Resultados del preentrenamiento

En el caso de detectar imágenes en las revistas, obtuvimos la siguiente matriz de confusión:

A partir de esta matriz, llegamos a la conclusión de que el modelo es bueno para identificar fotos verdaderas; sin embargo, arrojó algunos falsos negativos.

Con la matriz de confusión podemos calcular la precisión (*accuracy*):

$$\text{Precisión} = \frac{VP}{VP + FP} \tag{1}$$

$$\text{Precisión} = \frac{157}{157 + 35} = \frac{157}{192} = 0.82 \tag{2}$$

También tenemos los resultados de la métrica F1 (figura 6):

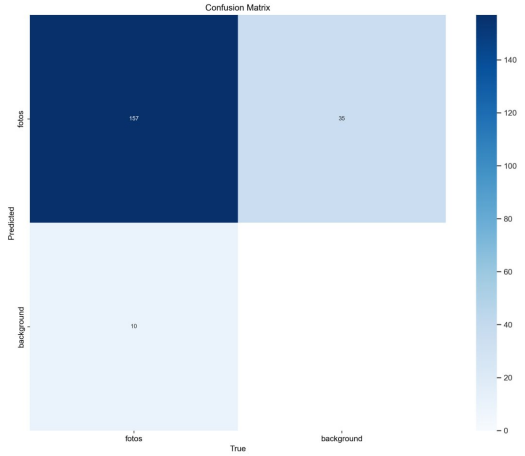

Figure 5: Matriz de confusión del modelo para detección de imágenes en revistas.

Este resultado indica que la curva alcanzó su puntuación máxima en 0.88 con un umbral de confianza de 0.55. Estos valores nos indican que, en general, el modelo es capaz de detectar la mayoría de los objetos.

Además, se midió también la métrica *Recall* (figura 7):

La gráfica indica que la curva tiene un *recall* máximo de 0.98 cuando el umbral de confianza es 0. Esto significa que, en promedio, el modelo es capaz de encontrar el 98% de todos los objetos cuando se fija un umbral de confianza de 0.

5.2 Evaluación del algoritmo de Supresión No Máxima (NMS)

También evaluamos el comportamiento del algoritmo de Supresión No Máxima (*Non-Maximum Suppression* - NMS), el cual es utilizado por YOLO para evitar detectar el mismo objeto (en este caso, imagen o *caption*) más de una vez en una foto.

Para esta evaluación, de las 25,000 fotos obtenidas, revisamos manualmente 1,300 imágenes, encontrando un total de 15 imágenes repetidas. Esto nos da una razón de:

$$\frac{15}{1300} = 0.011 \tag{3}$$

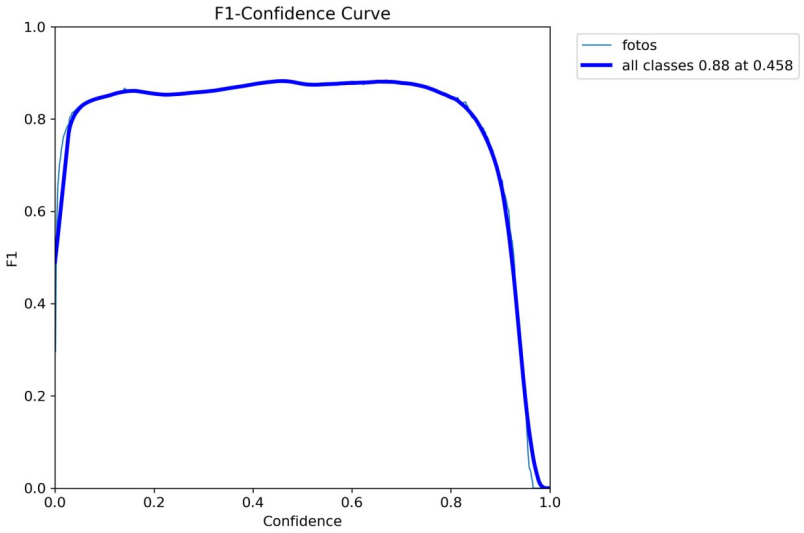

Figure 6: Curva F1 del modelo.

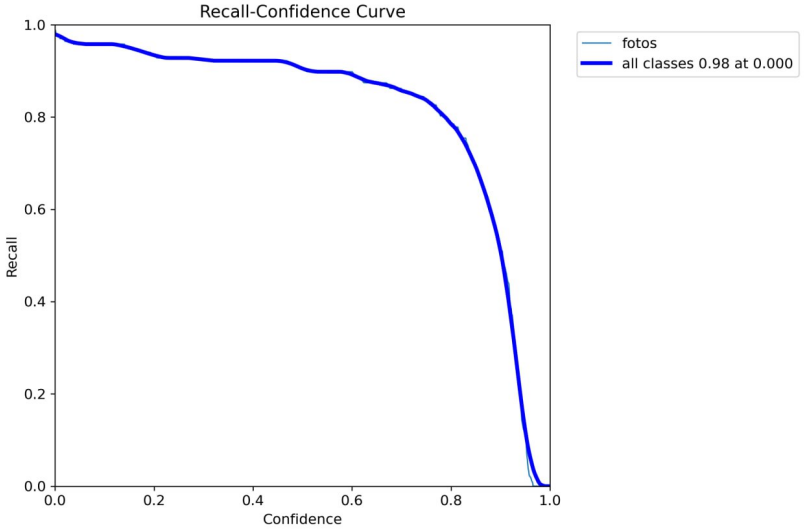

Figure 7: Curva de *Recall* del modelo.

Consideramos que este valor es bastante bajo, lo que indica un buen desempeño del algoritmo NMS en la supresión de detecciones redundantes.

5.3 Resultados de la limpieza

El proceso de depuración de datos inició con un conjunto de 5039 imágenes con sus respectivos *captions*. Como primer paso, se realizó un filtrado de ruido, eliminando 28 *captions* que presentaban problemas de caracteres invalidos. Posteriormente, se llevó a cabo un proceso de retoque y refinamiento de los *captions*, asegurando la coherencia y calidad de las descripciones sin necesidad de eliminar ningún elemento en esta fase.

Además, se evaluó la solidez de los datos utilizando el modelo *CLIP*, lo que llevó a la eliminación de 2000 *captions* que no describían adecuadamente las imágenes. Como resultado final de este proceso de filtrado y refinamiento, se obtuvo un conjunto depurado de 3011 imágenes con *captions*, garantizando una base de datos más precisa y confiable para el entrenamiento y evaluación del modelo.

5.4 Resultados del modelo

Como resultado del proceso de entrenamiento y validación del modelo, utilizando validación cruzada para distintos ajustes de hiperparámetros, se obtuvieron seis matrices de dimensión 10×50. Cada matriz representa la evaluación del modelo bajo un conjunto específico de hiperparámetros en 10 ejecuciones diferentes.

En este punto, la pregunta clave es: *¿Cuál de los ajustes proporciona el mejor rendimiento?* Para responder a esto, se llevó a cabo un análisis estadístico en dos fases:

1. Determinar si al menos uno de los ajustes es significativamente diferente a los demás.

2. Identificar si dicho ajuste es estadísticamente superior o inferior a los demás.

5.4.1 Análisis exploratorio de los datos

Para cada fila de una matriz, correspondiente a la evaluación del modelo en una base de datos específica, se calculó el promedio de sus valores. Esto

generó un vector de 10 componentes por cada configuración evaluada, obteniendo así un total de seis vectores.

El primer paso fue analizar si existía al menos un vector estadísticamente diferente a los demás. Inicialmente, se consideró la aplicación de un análisis de varianza (ANOVA). Sin embargo, este procedimiento requiere verificar el cumplimiento de tres supuestos fundamentales:

- **Independencia**: Se cumple, dado que cada *pipeline* se ejecuta de forma independiente, sin que los resultados de un experimento influyan en los demás.
- **Homogeneidad de varianzas**: Para evaluar esta condición, se analizaron los datos mediante diagramas de caja (*boxplots*) (Figura 8), los cuales sugieren que los *pipelines* 4 y 5 presentan una varianza distinta a la del resto. Para confirmar esta observación, se realizó una prueba de Levene, obteniendo un valor $p = 0.01118$, lo que indica que no se cumple la condición de homogeneidad de varianzas.
- **Normalidad**: Se verificó mediante histogramas de frecuencias y gráficos Q-Q (*Quantile-Quantile*) (Figuras 9,10,11,12). En particular, los puntos en el gráfico Q-Q del *pipeline* 4 muestran una desviación significativa de la línea de referencia, lo que sugiere una distribución no normal. Para corroborar esto, se aplicó la prueba de Shapiro-Wilk, obteniendo un valor $p = 0.0334$, lo que confirma que los datos no siguen una distribución normal.

5.4.2 Kruskal-Wallis y análisis Post Hoc

Dado que los resultados del análisis exploratorio de los datos arrojan que no se cumplen los supuesto de ANOVA, se optó por un análisis no paramétrico a través del test de Kruskal-Wallis. Esta prueba arrojó un valor $p = 2.77 \times 10^{-5}$, lo que indica que al menos un *pipeline* presenta una media significativamente diferente a las demás.

Posteriormente, se realizó un análisis *Post Hoc* para identificar cuál o cuáles *pipelines* mostraban diferencias significativas (Tabla 1). Los resultados indican que el *pipeline* 4 es significativamente distinto del resto.

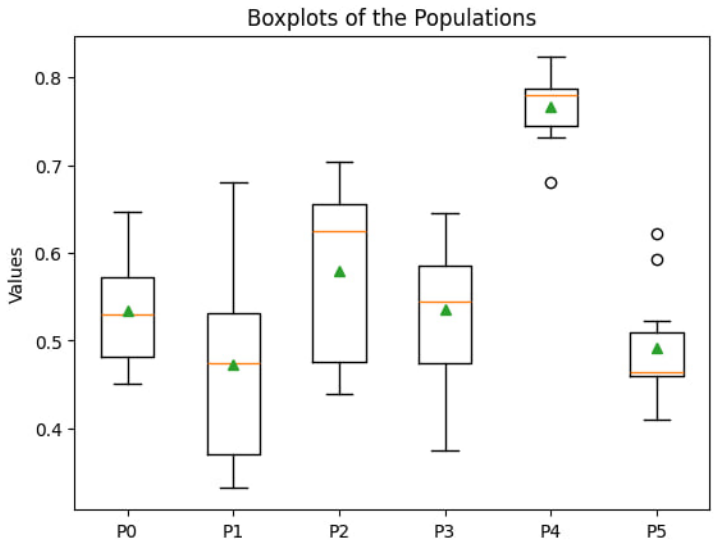

Figure 8: Distribución de los resultados de cada *pipeline* mediante diagramas de caja.

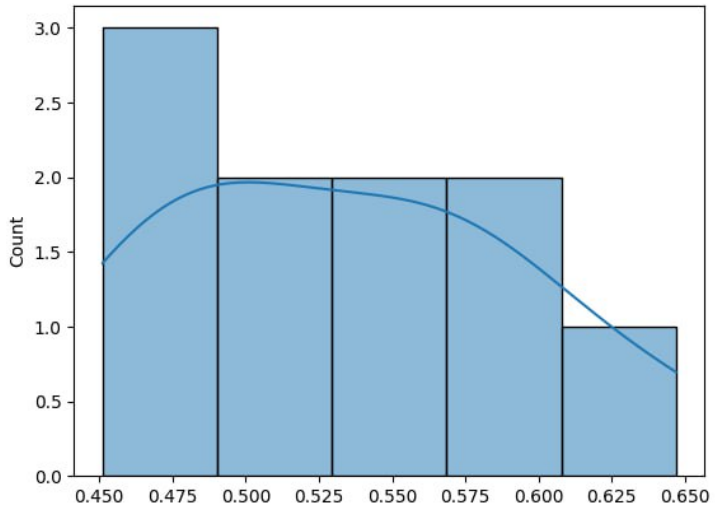

Figure 9: Histograma para el *pipeline* 0.

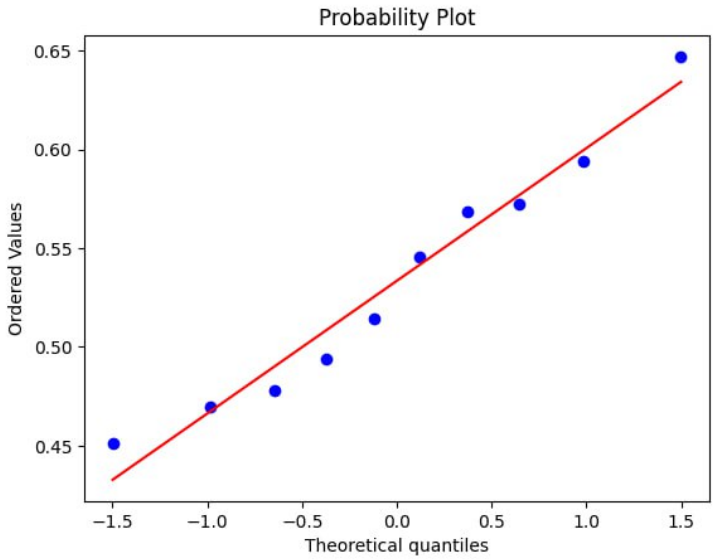

Figure 10: Gráfico Q-Q para el *pipeline* 0.

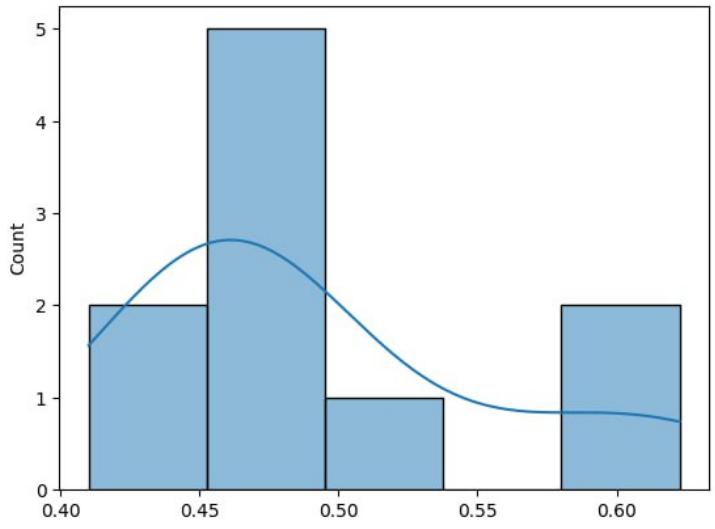

Figure 11: Histograma para el *pipeline* 4.

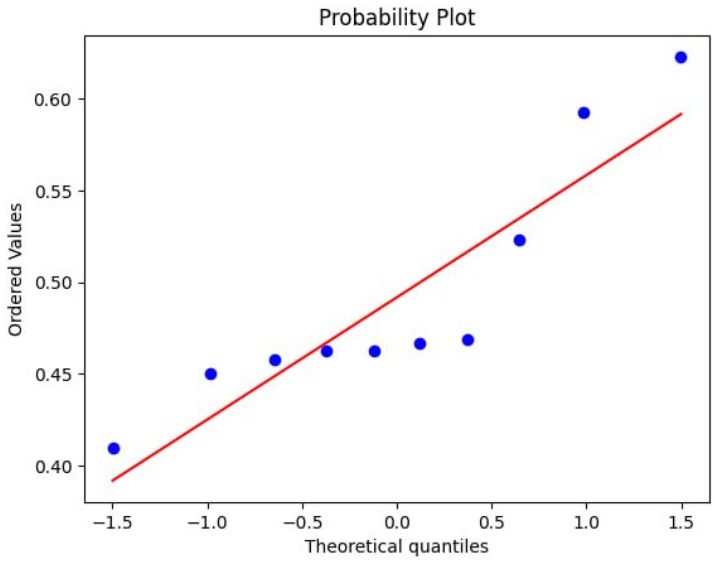

Figure 12: Gráfico Q-Q para el *pipeline* 4.

5.4.3 Análisis de rendimiento

Para evaluar si el *pipeline* 4 fue mejor o peor en términos de rendimiento, se calcularon los promedios de los valores de cada uno de los seis vectores obtenidos. Los resultados fueron los siguientes:

- p_0: Mean = 0.5334
- p_1: Mean = 0.4722
- p_2: Mean = 0.5789
- p_3: Mean = 0.5349
- p_4: Mean = 0.7673
- p_5: Mean = 0.4917

Dado que el *pipeline* 4 obtuvo la media más alta, se concluye que este ajuste, correspondiente a BLIP entrenado con los *captions*, superó a las demás configuraciones en términos de desempeño.

Pipeline Comparado	Valor p
p_4 vs. p_0	0.003
p_4 vs. p_1	0.001
p_4 vs. p_2	0.005
p_4 vs. p_3	0.002
p_4 vs. p_5	0.004

Table 1: Resultados del análisis *Post Hoc*, comparando el *pipeline* 4 con los demás.

6 Conclusiones y Trabajo Futuro

6.1 Resumen de resultados

A partir de los experimentos realizados y los análisis estadísticos efectuados, se obtuvieron las siguientes conclusiones:

- El modelo demostró un buen desempeño en la detección de imágenes en revistas, con una precisión del 82% y una puntuación F1 máxima de 0.88 para un umbral de confianza de 0.55. Esto indica que el modelo es capaz de detectar la mayoría de los objetos con una adecuada relación entre precisión y *recall*.

- La evaluación del algoritmo de Supresión No Máxima (NMS) mostró una tasa de detecciones repetidas del 0.01%, lo que sugiere un adecuado funcionamiento en la eliminación de predicciones redundantes.

- En el proceso de limpieza y filtrado de datos, se obtuvo un conjunto depurado de 3011 imágenes con *captions*, lo que garantiza una base de datos más precisa y confiable para entrenar y evaluar el modelo.

- El análisis comparativo de los distintos *pipelines* de entrenamiento evidenció que la configuración con BLIP entrenado con *captions* (denominada *pipeline* 4) obtuvo el mejor rendimiento, con una media de 0.7673, superando estadísticamente a las demás configuraciones según la prueba de Kruskal-Wallis y el análisis *Post Hoc*.

- Se verificó que los datos no cumplían los supuestos de normalidad y homogeneidad de varianzas, lo que llevó a utilizar pruebas estadísticas no paramétricas para evaluar las diferencias entre configuraciones.

6.2 Recomendaciones para futuros trabajos

Con base en los resultados obtenidos, se sugieren las siguientes recomendaciones para futuras investigaciones y mejoras del modelo:

- Explorar la integración de técnicas avanzadas de aumento de datos para mejorar la generalización del modelo y reducir la cantidad de falsos negativos detectados.
- Evaluar el impacto del ajuste de hiperparámetros en un espacio de búsqueda más amplio, considerando estrategias de optimización como *Bayesian Optimization* o *Hyperband*.
- Implementar un análisis más detallado del algoritmo NMS para entender su comportamiento en distintos umbrales de confianza y optimizar su funcionamiento.
- Comparar el desempeño del modelo entrenado con *captions* frente a otros enfoques basados en modelos de visión y lenguaje más recientes.
- Ampliar la base de datos de entrenamiento incorporando nuevas fuentes de imágenes y *captions*, asegurando diversidad en los datos para mejorar la robustez del modelo.
- Explorar la combinación de BLIP con otros modelos de generación de *captions* para evaluar sinergias y mejorar la precisión en la descripción de imágenes.

7 Referencias

- Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). *Show and Tell: A Neural Image Caption Generator*. CVPR.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need*. NeurIPS.
- Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, Ł., Shazeer, N., Ku, A., & Tran, D. (2018). *Image Transformer*. ICML.

- Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Amodei, D. (2021). *Learning Transferable Visual Models From Natural Language Supervision*. ICML.

- Li, J., Li, D., Xiong, C., & Hoi, S. C. (2022). *BLIP: Bootstrapped Language-Image Pretraining for Unified Vision-Language Understanding and Generation*. NeurIPS.

- Mokady, R., Bites, D., & Sadeh, T. (2021). *ClipCap: CLIP Prefix for Image Captioning*. arXiv preprint arXiv:2111.09734.

- Zhan, X., & Wu, Y. (2022). *Fine-grained Image Captioning with CLIP Reward*. Journal of Multimodal Intelligence, 3(2), 88-102. Retrieved from J-MIN.

- Li, Y., Lu, Z., & Wu, L. (2023). *BLIP-2: Bootstrapping Language-Image Pretraining with Frozen Image Encoders and Large Language Models*. Hackernoon. Retrieved from `https://hackernoon.com`.

- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You Only Look Once: Unified, Real-Time Object Detection*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779-788. `https://doi.org/10.1109/CVPR.2016.91`.