Universidad Zaragoza
1542

Trabajo Fin de Grado

Análisis del desajuste en las representaciones latentes multimodales con redes neuronales profundas

Modality gap influence over multimodal latent representations from deep neural networks

Autor

Andrei Calin Lazar Crisan

Directores

Eduardo Lleida Solano
Antonio Miguel Artiaga

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2024

AGRADECIMIENTOS

En primer lugar, quiero expresar mi profundo agradecimiento a mis tutores, Eduardo y Antonio, por su orientación y apoyo durante la realización de este Trabajo de Fin de Grado. Gracias a ellos, he podido ampliar mis conocimientos y mejorar mi enfoque hacia el trabajo, adquiriendo mayor madurez y rigor. Su ayuda ha sido fundamental para alcanzar estos logros.

Quiero dedicar unas palabras especiales de agradecimiento a mi familia, quienes han sido mi mayor apoyo durante los momentos más desafiantes de este camino académico. Su constante comprensión y sacrificio han sido fundamentales para que pueda llegar hasta mis metas. Agradezco de corazón todo el amor y el apoyo incondicional que me habéis brindado a lo largo de esta travesía.

Finalmente, muchas gracias a mis amigos. Han estado conmigo en los momentos buenos y en los malos, compartiendo risas, aventuras y dificultades. Cada experiencia vivida juntos nos ha acercado más y ha sido un recordatorio de lo afortunado que soy de tenerlos en mi vida. Gracias también a los compañeros de ViVoLab, que tanto en lo profesional como en lo personal, habéis hecho que todo esto haya sido más fácil.

RESUMEN

En la era digital actual, la cantidad de información generada y disponible ha crecido exponencialmente, lo que plantea un desafío significativo en términos de su recuperación y uso eficiente. Este desafío se intensifica cuando se trata de datos multimodales, es decir, aquellos que combinan texto, imágenes, audio y otros tipos de datos. La complejidad de estos datos multimodales requiere el desarrollo de técnicas avanzadas para obtener representaciones latentes de forma que sean recuperables.

Para enfrentar estos desafíos, se han desarrollado modelos y algoritmos específicos que permiten la creación de representaciones vectoriales de datos multimodales. En este trabajo, se utiliza *CLIP* (*Contrastive Language-Image Pre-training*), un modelo de red neuronal profunda que ha demostrado ser eficaz en la creación de representaciones vectoriales conjuntas para texto e imágenes. *CLIP* entrena simultáneamente sobre grandes cantidades de datos textuales e imágenes para aprender una representación conjunta en un espacio latente común.

Se analizan tres propiedades clave de las representaciones latentes obtenidas mediante *CLIP*: brecha intermodal, desalineamiento y agrupación. La brecha intermodal se refiere a la distancia que puede existir entre las representaciones de diferentes modalidades (por ejemplo, texto e imagen) para conceptos similares. El desalineamiento ocurre cuando las representaciones de las mismas entidades o conceptos no se superponen adecuadamente entre modalidades. La agrupación describe la tendencia de las representaciones a formar clústeres o agrupaciones en el espacio latente, lo cual puede ser beneficioso o perjudicial dependiendo del contexto.

En este trabajo, se evalúan diversos enfoques para mitigar el desajuste entre representaciones multimodales. Se exploran adaptadores específicos diseñados para mejorar la clasificación, así como modificaciones geométricas en la representación vectorial que buscan reducir las distancias intermodales. Estos métodos se analizan detalladamente para comprender su impacto en la alineación y eficacia de las representaciones latentes.

Además, se implementan técnicas avanzadas de optimización del entrenamiento para mejorar las representaciones latentes. Se desarrolla una herramienta de visualización que permite observar de manera geométrica el proceso de optimización, proporcionando una visión clara de cómo evolucionan las representaciones durante el entrenamiento. Los modelos se entrenan utilizando diversas funciones de coste y se trabaja con datos multilingües para evaluar la robustez de las representaciones.

Índice

1. Introducción **1**

1.1. Contexto 1

1.2. Objetivos 2

1.3. Planificación y desarrollo 3

1.4. Guía de la memoria 4

2. Los *embeddings*, el entrenamiento contrastivo y el modelo *CLIP* **5**

2.1. El concepto de *embedding* 5

2.2. Entrenamiento contrastivo 7

2.3. El modelo *CLIP* 10

2.3.1. Arquitectura 12

2.3.2. Proceso de entrenamiento 13

2.3.3. Aplicación a la clasificación y búsqueda 14

3. El desajuste multimodal **17**

3.1. La brecha intermodal 17

3.2. Métricas: desalineamiento y la agrupación 19

3.3. Técnicas de optimización del entrenamiento 20

3.3.1. Puntos de control de gradientes 20

3.3.2. Entrenamiento de las últimas capas 21

3.3.3. Adaptación de bajo rango mediante descomposición de pesos (*DoRA*) 22

4. Bases de datos **24**

4.1. *Microsoft COCO: common objects in text* 24

4.2. Pictogramas *ARASAAC* 25

4.3. Preprocesado de datos 26

5. Experimentación y análisis de resultados **28**

5.1. Visualización de la función de coste para un caso teórico 31

5.2. Discusión de resultados . . . 34
5.2.1. *MS-COCO* . . . 34
5.2.2. Pictogramas ARASAAC en inglés . . . 37
5.2.3. Pictogramas ARASAAC en español . . . 39

6. Conclusiones y líneas futuras **41**
6.1. Conclusiones . . . 41
6.1.1. Los espacios altamente dimensionales no se comportan de forma intuitiva . . . 42
6.1.2. Una función de coste más separable lleva a una representación menos transferible . . . 42
6.2. Líneas futuras . . . 43

7. Bibliografía **44**

Lista de Figuras **48**

Lista de Tablas **50**

Anexos **51**

A. Ejemplo de la base de datos de pictogramas de ARASAAC **52**

B. Revisión de soluciones **55**
B.1. Desplazamiento de vectores . . . 56
B.2. La consistencia intermodal e intramodal . . . 56
B.2.1. Adaptadores para clasificación . . . 56
B.2.2. *CyCLIP* y la consistencia intermodal . . . 59

C. Diseño de la interfaz de pruebas **61**

D. Resultados de los entrenamientos **63**
D.1. Pictogramas ARASAAC en inglés . . . 64
D.2. Pictogramas ARASAAC en español . . . 70

E. Los espacios altamente dimensionales no son intuitivos **76**

Capítulo 1

Introducción

1.1. Contexto

A lo largo de la historia hay una variable cuyo crecimiento es incuestionable: la información, en todas sus formas, manuscritos, arte, diagramas, fotografías y todas sus formas digitales. Más allá de la búsqueda manual, los primeros intentos de modernizar y automatizar la búsqueda de los elementos de un catálogo se realizaron por parte de *Emanuel Goldberg*, cuya colección de patentes incluye una máquina mecánica de búsqueda de patrones en un rollo de película. Continuado con la aproximación mecánica, Calvin Mooers [1] propuso en 1950 el uso de tarjetas perforadas con el fin de indexar los elementos de una colección. Fue en este momento cuando el término *information retrieval (IR)*, recuperación de información en español, fue creado y empleado ampliamente desde entonces.

Con el avance de la digitalización y la miniaturización de la electrónica, se comenzó a construir en la década de los 60 y 70 los primeros sistemas de búsqueda digital. Estos sistemas eran capaces de buscar sobre elementos indexados únicamente debido a las restricciones de memoria presentes.

Desde los años 80 hasta mediados de los 90 se popularizó una nueva aproximación: indexado a través de un modelo de espacio vectorial. La técnica más popular es el indexado semántico latente [2], mediante esta aproximación se determina en primer lugar la matriz de ocurrencia de la base de datos (matriz con las ocurrencias de una palabra en todos los documentos), posteriormente debido a su gran tamaño se realiza una aproximación de bajo rango (SVD) y la búsqueda se realiza en este espacio reducido con el vector de ocurrencias comprimido.

En la actualidad la búsqueda vectorial presenta una multitud de ventajas sobre su alternativa clásica. Gracias a los avances en el campo de la inteligencia artificial y técnicas de sensado comprimido, se permite la búsqueda rápida, eficiente y sobre todo multimodal. Esto quiere decir que el vector que representa un elemento en la colección

de datos, puede provenir de diversas fuentes y estas se podrán recuperar desde el resto de modalidades; por ejemplo, la búsqueda de imágenes a través de texto o de imágenes similares entre sí.

Una vez establecidas las bases y la historia de los sistemas de búsqueda y recuperación de información, se observa que presentan dos componentes principales: un codificador que convierte los datos provenientes de cualquier modalidad en su representación vectorial y, por otro lado, un sistema de búsqueda vectorial. En cuanto a la implementación de este último, no resulta trivial, sobre todo para sistemas de datos masivos. Este trabajo se centra en las propiedades deseadas en la representación vectorial para una correcta búsqueda y el efecto que tiene modificarlas. La aproximación que se lleva a cabo es la modificación del sistema codificador, así como la función de pérdidas de este durante el entrenamiento.

1.2. Objetivos

Los objetivos de este trabajo se centran en abordar los desafíos asociados con la representación eficiente y efectiva de datos multimodales utilizando el modelo *CLIP* . En primer lugar, el objetivo es realizar un entrenamiento eficiente y optimizado del sistema, aprovechando técnicas avanzadas para mejorar las representaciones vectoriales conjuntas de texto e imágenes.

En segundo lugar, se pretende profundizar en la comprensión de métricas específicas de desajuste multimodal como la brecha intermodal, el desalineamiento y la agrupación. Estas métricas son cruciales para evaluar la coherencia y la efectividad de las representaciones latentes obtenidas por *CLIP* , permitiendo identificar áreas de mejora.

Finalmente, se busca analizar y reducir el efecto del desajuste en las representaciones multimodales, especialmente considerando datos multilingües. Esto implica la exploración de adaptadores (sistemas adicionales de proyección), y modificaciones geométricas en las representaciones vectoriales para mitigar las discrepancias entre modalidades y mejorar la robustez de las representaciones frente a diferentes contextos lingüísticos.

En resumen, los objetivos principales incluyen el entrenamiento eficiente del modelo *CLIP* , la comprensión detallada de métricas de desajuste multimodal y la reducción del efecto del desajuste mediante técnicas avanzadas de optimización y adaptación de representaciones. Estos objetivos no solo contribuyen al avance teórico en el campo de las representaciones multimodales, sino que también tienen aplicaciones prácticas significativas en la mejora de sistemas de recuperación de información en entornos digitales multimodales.

1.3. Planificación y desarrollo

Se muestra en la figura 1.1, un diagrama de Gantt realizado a lo largo del proyecto. A continuación, se exponen brevemente las distintas fases del trabajo realizado. El primer paso para un trabajo de investigación es la revisión de la literatura existente del problema en cuestión. Para este trabajo ha sido el proceso con mayor dedicación sin duda alguna, esto se debe a dos motivos, el primero es la necesidad de documentarse hasta ponerse al día tanto con los sistemas como con las técnicas y aproximaciones mencionadas. Por otro lado, teniendo en cuenta el gran ritmo que existe hoy en día en cuanto a creación de artículos científicos, sobre todo en un campo en auge como el de la inteligencia artificial, se requiere de un esfuerzo continuo para mantenerse actualizado con el campo o para transferir técnicas utilizadas para resolver problemas semejantes.

Una vez obtenida la información y determinado el sistema que se va a realizar, se construye un entorno de desarrollo en el que se permiten hacer las pruebas pertinentes para entrenar el sistema. Posteriormente, se deben de seleccionar las bases de datos a utilizar y caracterizarlas. A posteriori, con el entorno de desarrollo construido se realizan optimizaciones al sistema con el fin de minimizar los recursos necesarios y finalmente se realizan diversos entrenamientos con las modificaciones propuestas. Finalmente, se procede al análisis de resultados y presentación de los mismos.

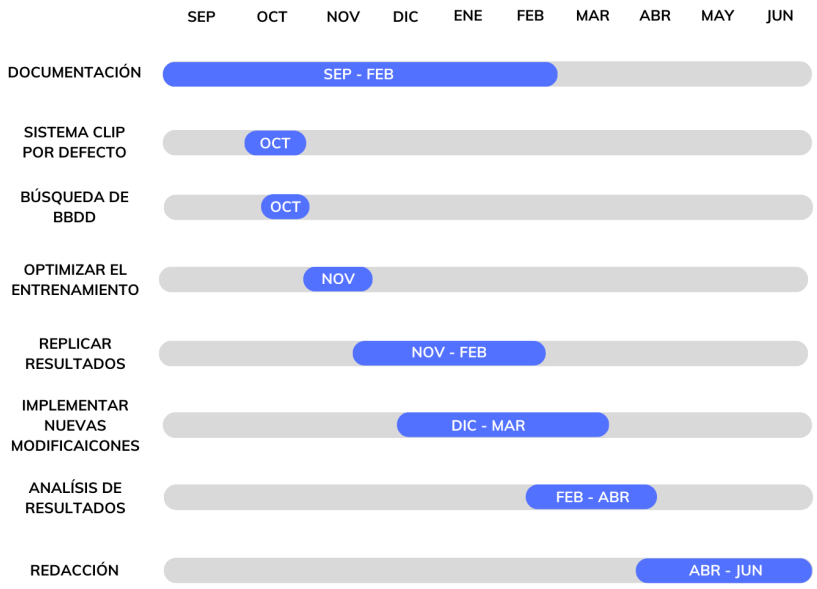

Figura 1.1: Diagrama de *Gantt* para el desarrollo del proyecto.

1.4. Guía de la memoria

Esta memoria se encuentra compuesta por cinco capítulos y cinco anexos, a continuación, se procede a enunciar brevemente el contenido de cada uno de ellos.

Capítulo 1. Introducción: En este capítulo se presenta un breve resumen de la memoria, la planificación temporal y la presente guía.

Capítulo 2. Los *embeddings, el entrenamiento contrastivo y el modelo CLIP*: Este capítulo presenta el concepto de *embedding* o representación latente, el modelo usado, *CLIP* y el entrenamiento no supervisado contrastivo.

Capítulo 3. El problema del desajuste multimodal: Una vez se conoce el funcionamiento del sistema propuesto para búsqueda multimodal de información, se presenta el problema de la brecha o desajuste multimodal. Se expone su origen, cómo cuantificarlo y las técnicas que se han utilizado para solventarlo. Adicionalmente, se muestran las técnicas implementadas para mejorar la eficiencia de recursos durante el entrenamiento.

Capítulo 4. Bases de datos: En este capítulo se exponen y caracterizan las bases de datos empleadas para validar los resultados.

Capítulo 5. Experimentación y análisis de resultados: Una vez descrito el problema y elegidas las bases de datos, se procede a la fase experimental. Primero, se realiza un pequeño experimento en el que se visualizan las funciones de coste para distintas distribuciones espaciales, con una finalidad de aumentar la intuición sobre el problema. Posteriormente, se replican resultados de otras aproximaciones, pero haciendo uso de estas bases de datos y posteriormente se toma una nueva visión al problema proponiendo funciones de coste alternativas.

Capítulo 6. Conclusiones y líneas futuras: Finalmente, en este capítulo se da un paso atrás y se intenta tomar un punto de vista lejano con el fin de extraer las conclusiones del trabajo realizado y contextualizarlos en el marco de los resultados existentes.

Capítulo 2

Los *embeddings*, el entrenamiento contrastivo y el modelo *CLIP*

2.1. El concepto de *embedding*

El concepto de *embedding* o vector de representación hace referencia a una representación vectorial asociada a un dato. Esta representación requiere de forma general un espacio vectorial cuya dimensión es mucho menor a la del dato, un codificador que permite la codificación del dato y finalmente una medida de similitud en este nuevo espacio vectorial latente o comprimido.

Los primeros sistemas de extracción de características comenzaron en la década de los 50 para sistemas de indexado y búsqueda en documentos de textos. Las primeras aproximaciones se basan en el conteo de la frecuencia relativa de cada palabra en un documento [3]. De esta forma, cada dimensión del vector representa la frecuencia relativa de una palabra en el documento y por consecuencia, se tiene que documentos con frecuencias relativas similares para un subconjunto de palabras se encuentran agrupados entre sí y distantes del resto. Esta aproximación presenta una gran desventaja: la frecuencia relativa de las palabras en un idioma no es constante. Por ello se propone posteriormente la ponderación de la frecuencia relativa en todo el idioma [4].

El mayor salto en capacidad de modelado de lenguaje se produce en 2013 con la aparición del modelo *Word2Vec* [5], mediante el uso de redes neuronales. La idea reside en predecir la siguiente palabra dado un texto de entrada. Se tiene una ventana deslizante que recorre el texto, en cada paso se intenta predecir la palabra central con las palabras cercanas a modo de contexto [1]. Estos vectores, pese a ser entrenados con el esquema propuesto anteriormente, la representación presenta dos propiedades a destacar: la primera es la agrupación semántica de palabras. En ningún momento,

[1] El método descrito se conoce como *continuous bag of words*, las palabras se predicen con una ventana continua.

se entrena de forma directa la asociación de palabras semánticamente similares en regiones cercanas del espacio vectorial, pero explotando las cualidades estadísticas de texto se consigue. Por otro lado, se tiene que los vectores responden a una cierta aritmética vectorial lineal; p. ej. $(E[rey] - E[hombre]) + E[mujer] \approx E[reina]$, donde E representa el vector de características o *embedding* asociado a la palabra.

Hasta este momento las técnicas descritas se centran en el modelado de lenguaje, no se trata de una coincidencia que los primeros avances en la representación o búsqueda de información, hayan sido en esta modalidad. El texto o lenguaje, por su propia naturaleza discreta, facilita el proceso de incrustación u obtención de una representación vectorial. Se considera el texto como una modalidad de información discreta debido a que su unidad elemental, el grafema, forma parte de un conjunto discreto. A partir del grafema, la primera unidad que posee significado es la palabra, esta a su vez se compone de una raíz o lexema y un morfema [2], de esta forma muchas de las cualidades que se desea en la representación vectorial se obtienen por la propia naturaleza del lenguaje. La modalidad de la imagen no tiene esta suerte. A su pesar la información que contiene por lo general es mayor que una descripción asociada a ella, la extracción de la semántica a partir de la imagen se convierte en una tarea más dura. Esto no implica que en alguna de las dos modalidades exista información que no esté presente en el resto, en ese caso no carece de sentido considerar las distintas modalidades debido a que en ese caso con una única se tiene toda la información necesaria.

Una vez expuesto el concepto de *embedding* o vector de representación, es intuitivo desear obtener una representación vectorial del conocimiento a través de distintas modalidades. En el marco de este trabajo se consideran las modalidades de texto e imagen. Se desea una representación conjunta de forma que ambos vectores se encuentren cercanos en el espacio de representación conjunto. Hasta ahora, se ha hecho amplio uso de las palabras *cercano* y *representación conjunta*, pero su interpretación no presenta una solución trivial ni generalizada para cualquier problema de representación de información. En primer lugar, se debe de definir una distancia o norma en el espacio vectorial, de esta forma se tienen distancias pequeñas para representaciones semánticamente similares y viceversa. Finalmente, queda por determinar el mecanismo de obtención de la representación conjunta, este depende de diversos factores: la disponibilidad de datos anotados, la función de coste, la dimensión del vector, etc.

[2] Un ejemplo de la estructura elemental semántica de las palabras se tiene al considerar como lexema *pequeñ-* y añadiendo morfemas gramaticales como sufijos se forman las siguientes palabras *-a; pequeña* o *-ito; pequeñito*.

2.2. Entrenamiento contrastivo

En el ámbito de aprendizaje automático existen tres paradigmas de aprendizaje: en primer lugar, el aprendizaje supervisado, que tiene como objetivo predecir las etiquetas proporcionadas durante el entrenamiento. Estas etiquetas pueden ser clases, en ese caso se trata de un problema de clasificación, por lo contrario, si se tiene etiquetas numéricas se trata de un problema de regresión. El aprendizaje no supervisado, presenta como característica principal la innecesariedad de etiquetas para los datos del entrenamiento. En este caso, el algoritmo busca explotar algún tipo de estructura presente en los datos, identificando las similitudes entre diferentes características de los datos. En este caso, el resultado del modelo puede ser una agrupación o bien un vector o *embedding* de dimensión reducida con respecto a la entrada. Así, se tiene que estas nuevas dimensiones han condensado las diferencias entre ejemplos. Finalmente, se tiene el aprendizaje reforzado, para aplicar este paradigma se tienen que dar las siguientes condiciones: en primer lugar, poseer de un conjunto de estados de entorno $\boldsymbol{S}$ mediante el cual interactúa el agente, un conjunto de acciones $\boldsymbol{A}$ que puede tomar y finalmente las reglas de recompensa $\pi(s)$, tanto asociadas a la transición entre estados como a la recompensa inmediata. De esta forma se trata de maximizar la recompensa asociada a un estado y una acción. Este tipo de técnicas se utilizan ampliamente para resolver problemas de optimización cuyos límites son extremadamente grandes, por ejemplo motores de evaluación o juego de ajedrez, optimización en problemas de logística o automatización de robots, entre otros. El gran problema que presenta esta aproximación es la dificultad de cómputo o la simulación del entorno junto con el agente, por ejemplo si se desea simular un robot interactuando en un escenario virtual, para una sola acción se deben de simular todas las colisiones asociadas al motor de física del entorno además del propio modelo del agente y la constante computación de la función de coste.

Para la aplicación de las diversas técnicas de aprendizaje automático es crucial el conocimiento del problema que desea abordar. En la Figura 2.1 se detalla, a grandes rasgos, un proceso de identificación de la categoría de aprendizaje automático a utilizar. Para ello es necesario conocer el resultado que se quiere obtener junto a la disponibilidad de los datos.

La supervisión en un sistema hace referencia a la capacidad o disposición de las etiquetas asociadas a cada dato. Hasta ahora se ha presentado como una cualidad que existe o no, pero no es así, de hecho existe un gran espectro de supervisión en diversos problemas. Con el fin de ilustrar los diversos grados de supervisión, se considera un problema de clasificación de imágenes de animales. Suponiendo que se posee un conjunto de imágenes en el que aparen únicamente 10 especies distintas. Se

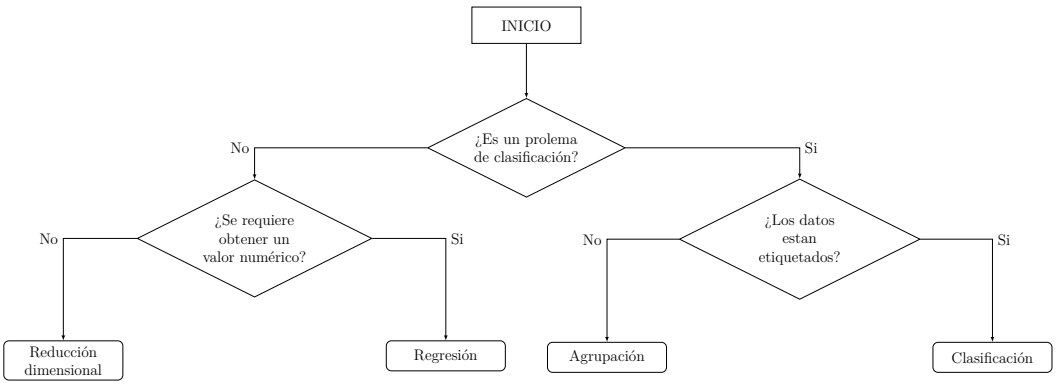

Figura 2.1: Clasificación de los paradigmas de aprendizaje automático.

tiene que se podrá entrenar un modelo, que dada una nueva imagen, sea clasificada entre las 10 especies con las que ha sido entrenado. Este es el ejemplo clásico de aprendizaje completamente supervisado. En el otro extremo, se tiene únicamente un conjunto de imágenes sin su correspondiente etiqueta. Mediante algoritmos de agrupación se consigue explotar atributos estadísticos de las imágenes y tras un proceso de agrupación, se observa que el algoritmo ha conseguido realizar la *clasificación*[3] entre animales terrestres y acuáticos. Este es un ejemplo de un paradigma de aprendizaje no supervisado. Finalmente, a continuación se considera que se tienen parejas de imágenes junto con pies de foto, obtenidos a partir de un proceso de búsqueda en revistas de naturaleza. En este caso no se tiene un problema de clasificación dado que no existe un número predeterminado de etiquetas. Por el contrario, si se trata como un problema no supervisado, se estaría perdiendo la información que puede añadir las correspondientes descripciones o pies de foto. En este caso la supervisión no es nula ni total, a este tipo de problemas también se les conoce con el nombre de supervisión débil o propia.

El aprendizaje mediante supervisión propia (*SSL*[4]), tal y como se expone en [6], se puede realizar desde dos aproximaciones distintas: los métodos contrastivos y los no contrastivos. A continuación, se detallan exponiendo su principio de funcionamiento, así como diversos paradigmas existentes de cada uno.

[3]No debe considerarse como un problema de clasificación como tal, dado que no se tienen etiquetas explícitas y, por lo tanto, el modelo desconoce el objetivo deseado.

[4]Del inglés: *Self Supervised Learning*, traducido como aprendizaje mediante supervisión propia.

– ***SSL* Contrastivo:** Sea $\{x_i\}$ un conjunto de muestras de entrada de forma que cada una de ellas posee su correspondiente etiqueta $y_i \in \{1, 2, ..., L\}$, entre las L posibles clases. Se desea aprender una función $f_\theta : X \to \mathbb{R}^d$ tal que codifica x_i en un vector de representación d dimensional, de forma que se consigue que ejemplos pertenecientes a la misma clase posean vectores *similares* y viceversa. Por lo tanto, el entrenamiento contrastivo parte de una pareja de ejemplos (x_i, x_j) y minimiza la distancia entre los *embeddings* cuando pertenecen a la misma clase y en caso contrario la maximiza. En este caso se define la siguiente función de pérdidas:

$$\begin{aligned}\mathcal{L}_{cont}(x_i, x_j, \theta) =& \mathbb{1}[y_i = y_j]\|f_\theta(x_i) - f_\theta(x_j)\|_2^2 + \\ & \mathbb{1}[y_i \neq y_j]\,\text{máx}\{0, \epsilon - \|f_\theta(x_i) - f_\theta(x_j)\|_2^2\}\end{aligned} \tag{2.1}$$

Donde $\mathbb{1}[y_i = y_j]$ representa una matriz con unos en los elementos $y_i = y_j$ y de forma similar, $\mathbb{1}[y_i \neq y_j]$ en los índices disjuntos. Adicionalmente, se introduce el hiperparámetro ε, de forma que representa una cota inferior de la distancia entre parejas no correspondientes.

La mayor dificultad de los métodos contrastivos reside en la búsqueda de los ejemplos negativos. Para ello habitualmente se hace uso de técnicas de aumento de datos, por ejemplo con las imágenes se realizan rotaciones y recortes aleatorios. En el caso de que se disponga de ejemplos negativos anotados manualmente, el resultado será mejor que mediante técnicas de aumentado de datos, en este caso el nivel de supervisión aumenta. Un ejemplo de este tipo de aproximación se tiene en [7], partiendo de fotografías de rostros se entrena de forma contrastiva una red neuronal convolucional, de forma que los ejemplos pertenecientes a la misma persona se encuentran cercanos en el espacio latente. A diferencia de haberlo entrenado de una forma totalmente supervisada, como un problema de clasificación, se tiene que mediante el entrenamiento contrastivo la red neuronal aprende una codificación que maximiza la separabilidad entre rostros. Por lo tanto, aunque no haya sido entrenado en un rostro en particular, se puede obtener la representación vectorial asociada a este. A posteriori, mediante una métrica, determinar la similitud entre ejemplos sin necesidad de volver a entrenar el modelo. A esta categoría pertenecen los algoritmos como *InfoNCE* [8] o *SimCLR* [9], los cuales actualmente suponen las bases del entrenamiento contrastivo.

– ***SSL* No-Contrastivo:** A diferencia del entrenamiento contrastivo, las técnicas no contrastivas únicamente hacen uso de los ejemplos positivos. Pese a que existen muchas aproximaciones, todas ellas parten de la misma idea: dado una pareja de ejemplos, estos se codifican a un espacio latente de forma similar a los métodos contrastivos, a diferencia de estos no se calcula la similitud entre ellos, sino que se realiza una proyección a un espacio dimensional de mayor dimensión[5] en el que sí que se calcula la métrica entre proyecciones. Este tipo de técnicas presentan problemas de convergencia de forma que la representación latente colapsa a cero, de esta forma la función de coste (distancia entre vectores) se minimiza, por lo que se requieren de técnicas de regularización que permitan la convergencia a un mínimo local y no el global trivial. A esta categoría pertenecen métodos como: *Barlow-Twins* [10] o *I-Jepa* [11], en el ámbito multimodal.

2.3. El modelo *CLIP*

Antes de detallar el modelo *CLIP*, se procede a explicar atentamente el modelo *ConVIRT* [12] dado que la mayoría de ideas de *CLIP* proceden de este. En el caso de *ConVIRT*, se pretende obtener una representación conjunta de imágenes médicas junto con diagnósticos, de forma que los vectores densos puedan ser utilizados para tareas posteriores como clasificación o búsqueda de imágenes a partir de descripciones textuales. En primer lugar, la aproximación utilizada es de tipo contrastivo, inspirado en el reciente éxito de este tipo de métodos, tal y como se expone en *Representation learning with contrastive predictive coding* [8] y *SimCLR* [9].

Partiendo de un conjunto de imágenes $\mathbf{x_v}$ emparejadas con sus respectivas descripciones $\mathbf{x_u}$, se pretende aprender una función parametrizada $\mathbf{f_v}$ de forma que se asigna a cada imagen un vector $\mathbf{h_v} \in \mathbb{R}^d$, de la misma forma se define un codificador de texto $\mathbf{f_u}$ que transforma cada descripción a un vector $\mathbf{h_u} \in \mathbb{R}^d$. A cada una de estas representaciones se les aplica finalmente una transformación no lineal g_u y g_v de forma que se tienen los vectores $\mathbf{v}$ y $\mathbf{u}$, a estos se les aplica la función de coste contrastiva tal que las parejas similares se encuentren cerca y viceversa.

Debido a la escasez de datos etiquetados de aplicación médica, se propone el uso de técnicas de aumento de datos o transformaciones que se aplican a los datos de entrada. Para el caso de las imágenes, se tiene una distribución de posibles transformaciones, $\mathcal{T}$ de forma que la imagen de entrada al modelo es la transformación de dicho dato $\tilde{\mathbf{x}}_\mathbf{v} = t_v(\mathbf{x_v})$. La descripción textual se obtiene muestreando de forma uniforme una

[5] Esta dimensión intermedia es mayor que la del vector denso llegando a ser en ocasiones la inicial, en ese caso se tiene un proceso de compresión-reconstrucción.

frase perteneciente al informe médico asociado a dicha imagen, de esta forma se tiene $\tilde{\mathbf{x}}_\mathbf{u} = t_u(\mathbf{x}_\mathbf{u})$. Para el caso de las transformaciones de las imágenes se muestrean de forma secuencial y aleatoria imágenes resultantes de las siguientes modificaciones: *recorte*, *volteado horizontal*, *transformación afín*, *ajuste de brillo* y *desenfoque gaussiano*. En la figura 2.2, se presenta el esquema de la arquitectura de *ConVIRT* con la nomenclatura expuesta anteriormente.

Hasta ahora se ha hecho referencia a los codificadores de imagen y texto de una forma genérica, en el caso de *ConVIRT*, se tratan de arquitecturas basadas en redes neuronales profundas. Para la modalidad del texto, se hace uso de únicamente la parte codificadora de *BERT*[13], una arquitectura de tipo *transformer*. Por otro lado, para la modalidad de imagen, se hace uso de una red neuronal convolucional en concreto, tipo *ResNet50* [14]

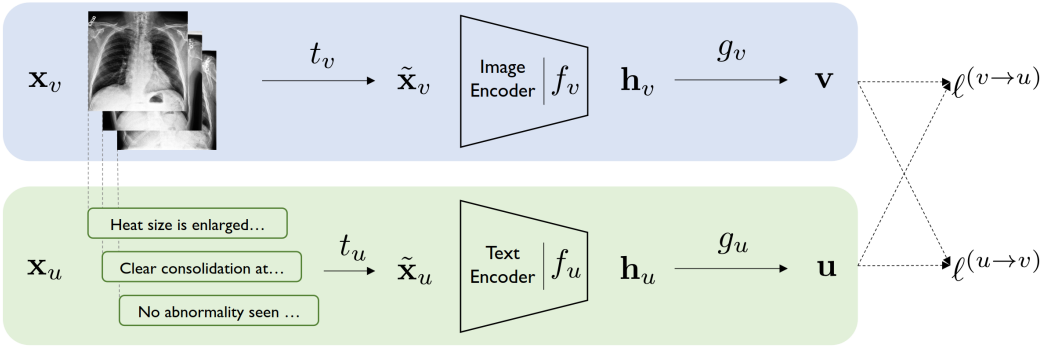

Figura 2.2: Diagrama de la arquitectura *ConVIRT*. [6]

La arquitectura de *CLIP* se encuentra ampliamente inspirada en la descrita anteriormente, se simplifica ligeramente tanto la arquitectura como el proceso de aumento de datos, gracias a la amplia disponibilidad de parejas de imagen y texto de carácter general. Finalmente, aprovechando los últimos desarrollos en modelos de visión, se introduce una alternativa basada en *transformer* para el codificador de imagen.

[6] Fuente: Zhang, Y. et al. [12].

2.3.1. Arquitectura

Tal y como se ha comentado anteriormente, en el caso de *CLIP* [15] se produce una simplificación de la arquitectura. En primer lugar, se elimina la última capa de proyección no lineal, esta no tiene mucho sentido si se posee una red con suficientes capas, de esta forma las tareas de proyección se trasladan a la propia red. Por otro lado, se simplifican las transformaciones a considerar únicamente recortes en la parte de imagen. En la modalidad de texto se introduce directamente la descripción asociada a la imagen gracias a que estas son mucho más cortas, aun así, se limita la longitud del texto a 77 *tokens*. Finalmente, se entrena adicionalmente el valor de temperatura τ asociado a la capa softmax de cada modalidad, esto se puede interpretar como una pseudo-calibración entre modalidades.

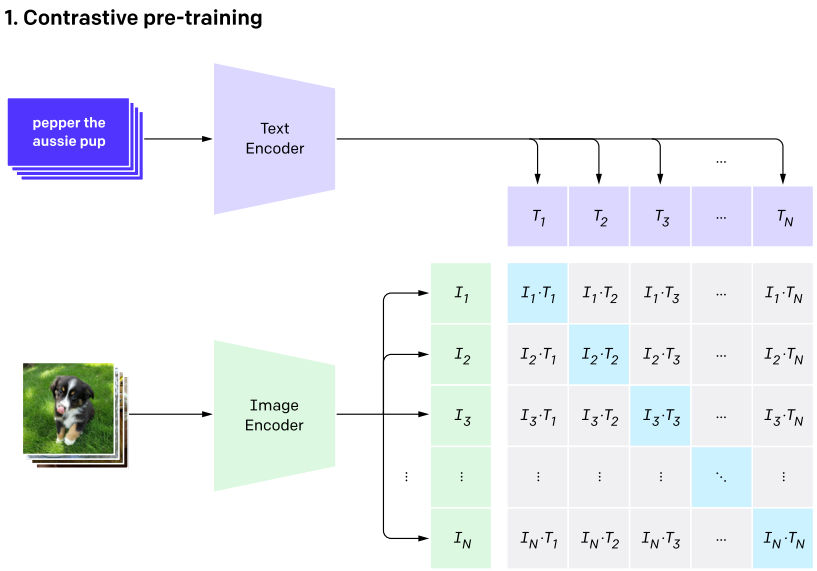

Figura 2.3: Diagrama de la arquitectura de *CLIP*.[7]

Para *CLIP* se define la similitud como el producto coseno[8], de esta forma, considerando que los vectores densos se encuentran normalizados mediante la norma euclídea ($\|\cdot\|_2$), este es equivalente a realizar el producto escalar entre los vectores. El resultado de aplicar el producto coseno a todos los elementos de un lote es una matriz cuadrada simétrica, de forma que los elementos de la diagonal contienen la similitud coseno entre las parejas correctas y el resto entre los pares cruzados. En la figura 2.3 se observa la creación de esta matriz a partir de imágenes y textos. De esta forma, se pretende maximizar el valor de la diagonal y a su vez alejar el resto, para ello se hace uso de la entropía cruzada de cada fila (búsqueda mediante texto) y columna (búsqueda mediante imagen), utilizando como etiquetas los índices de la diagonal.

[7] Fuente: OpenAI [15]

[8] Se define el producto coseno como el producto escalar normalizado: $\langle u, v \rangle = \frac{u \cdot v}{\|u\| \cdot \|v\|}$

2.3.2. Proceso de entrenamiento

De forma general, a continuación, se presenta el algoritmo de entrenamiento para un sistema como *CLIP*. En primer lugar, se obtienen los vectores densos a partir de cada modelo, estos vectores se ven normalizados mediante la norma euclídea para encontrarse en una esfera d-dimensional ($d = 768$ para el modelo de *CLIP* en cuestión), de esta forma el producto escalar presentará valores contenidos entre -1 y 1. A partir de los *logits*, definidos en el paso 5 del algoritmo 1, se obtienen las probabilidades predichas mediante la función softmax. Estos *logits* hacen de valores predichos en la función de coste, de esta forma se desea que el producto a maximizar sea la pareja asociada a la diagonal. En este caso, se definen dos términos en la función de pérdidas, cada uno de ellos asociado a la comparación entre el ejemplo correcto de la modalidad de texto con todas las imágenes de la tirada y viceversa, esto se traduce a considerar la matriz transpuesta de *logits*.

La elección de la entropía cruzada como función de coste no se realiza de una forma arbitraria, esta se deriva de la expresión óptima para un criterio de maximización de verosimilitud en un clasificador bayesiano. En el caso que se estudia, no se trata de un problema de clasificación clásico; sin embargo, dado que se considera una única pareja como la correcta en una tirada se puede ver como un problema con un número de clases igual al tamaño del lote y cuyo único ejemplo positivo es la pareja de imagen y texto correspondiente. De esta forma, se tiene $CE = -\sum_{i=1}^{N} y_i \log p(y = i|x)$ con y_i la etiqueta real y `softmax`(*logits*) la predicha por el modelo. A continuación, se muestra un algoritmo que contiene la implementación de *CLIP* en pseudocódigo.

Algoritmo 1 Implementación en pseudocódigo de *CLIP*

1: $I_f \leftarrow$ codificador_imagen(imagen)
2: $T_f \leftarrow$ codificador_texto(texto)

3: $I_{norm} \leftarrow \|I_f\|_2$
4: $T_{norm} \leftarrow \|T_f\|_2$

5: Logits $\leftarrow \langle I_{norm}, T_{norm} \rangle \cdot e^{\tau}$ ▷ $\langle u, v \rangle$ denota el producto escalar entre u y v

6: $Etiquetas \leftarrow [1, 2, 3, ..., N]$
7: $\mathcal{L}_i \leftarrow CE\left[Logits, Etiquetas\right]$
8: $\mathcal{L}_t \leftarrow CE\left[Logits^T, Etiquetas\right]$
9: $\mathcal{L} \leftarrow (\mathcal{L}_i + \mathcal{L}_t)/2$

Actualmente, la mayoría de modelos se entrenan mediante el uso de procesadores gráficos, debido a la necesidad de realizar operaciones vectoriales de una forma eficiente. Dicho esto, existen limitaciones tanto con la memoria disponible como la velocidad

de procesado. El mecanismo de entrenamiento descrito en el algoritmo 1, presenta el mayor cuello de botella en la memoria requerida para calcular los vectores densos. Los modelos una vez cargados en memoria no requieren más de 2 GB, considerando que todos sus pesos y gradientes se encuentran en el tipo de dato `float32`. La mayor parte de la memoria se requiere para cargar las imágenes y los gradientes asociados a ellas, el aumento del número de datos en una iteración es prácticamente lineal por lo que con los recursos disponibles (24 GB), adicionalmente, mediante el uso de técnicas de optimización descritas en el capítulo 3.3, se logra tener un tamaño de lote máximo de 90 imágenes y descripciones.

Para tareas clásicas como el entrenamiento de un modelo para clasificación mediante aprendizaje supervisado, la limitación del tamaño de lote no supone un gran problema, en el caso del entrenamiento contrastivo, es un parámetro crucial. Dado que el entrenamiento contrastivo se basa en la diferencia entre ejemplos, el cálculo del gradiente entre ellos se realiza únicamente si pertenecen al mismo lote.

2.3.3. Aplicación a la clasificación y búsqueda

La gran ventaja de este tipo de modelo es la capacidad de generalización a través de un entrenamiento previo, para ello *CLIP* ha sido entrenado con una variedad de conjuntos de datos entre los que se incluyen *MS-COCO* [16]. La capacidad de aprovechar el conocimiento previo junto con pocos ejemplos o ningunos de la nueva tarea recibe el nombre de *few-shot* o *zero-shot* respectivamente, en este aspecto *CLIP* representa un gran salto debido a la versatilidad que ofrece en un amplio abanico de tareas.

Ahora bien, si se desea hacer uso de *CLIP* como un clasificador, se han de calcular los *logits* descritos anteriormente y aplicar la transformación `softmax`: $\sigma(x_i) = \frac{e^{x_i}}{\sum_{j=i}^{N} e^{x_j}}$, de esta forma se logra transformar un vector de puntajes a un vector de probabilidades para cada clase. En la figura 2.4 se muestra un ejemplo de la clasificación de imagen, así como la predicción de texto a partir de la imagen (clasificación de texto).

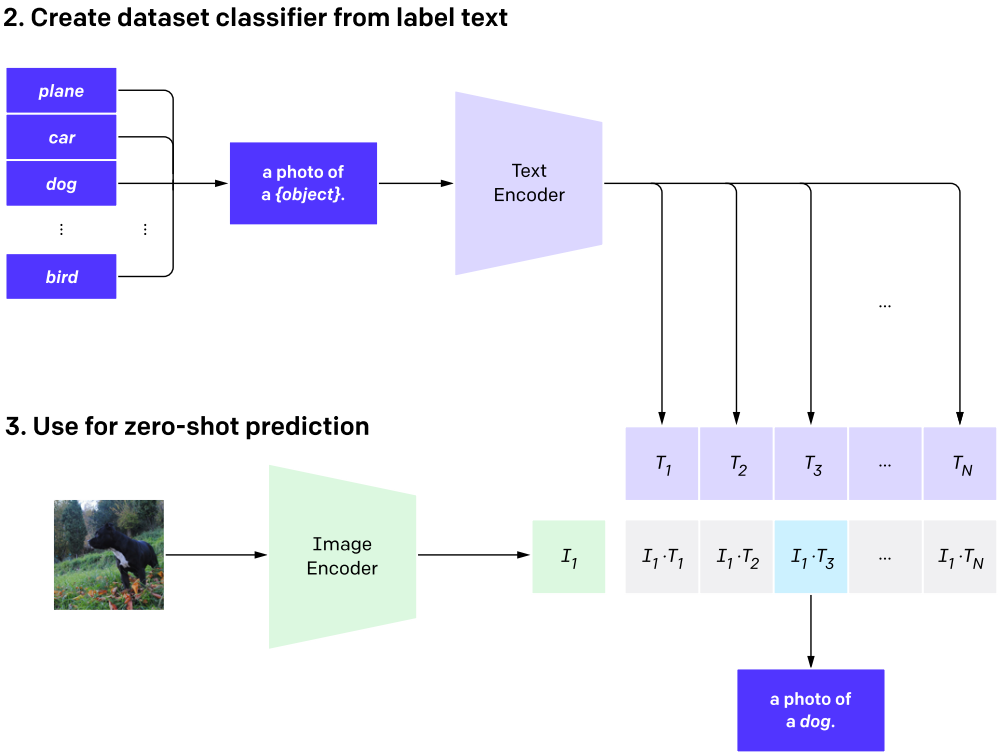

Figura 2.4: Diagrama de clasificación mediante *CLIP*.[9]

Por otro lado, se tiene la tarea de recuperación o búsqueda de información, para ello, en lugar de aplicar la transformación `softmax`, se trabaja con el valor del producto coseno directamente, el escalado por temperatura es una constante que a la hora ordenar los resultados resulta irrelevante. Para esta tarea, se definen las métricas de *retrieval* $R@k$, con k un entero positivo, esta métrica mide el porcentaje de veces que el resultado relevante se encuentra entre los k primeros elementos. Un factor crucial para esta métrica es el tamaño de la muestra, para muestras con un número reducido de elementos, se tienen valores elevados con mayor facilidad porque los k primeros elementos suponen una parte significativa de la muestra. Por el contrario, un tamaño de muestra elevado dificulta el valor de $R@k$, sobre todo si se tienen ejemplos similares. Finalmente, para comparar distintos valores de $R@k$ es crucial compararlos con el mismo tamaño de muestra y esta práctica se asegura durante todo el trabajo estandarizándolo al mismo valor que el tamaño de lote: 90 muestras.

[9]Fuente: OpenAI [15]

A continuación, se presenta un esquema de diseño para una aplicación de búsqueda vectorial en la figura 2.5. El principio de funcionamiento es sencillo: se obtienen las representaciones vectoriales para todas las entradas de la base de datos y para realizar el proceso de búsqueda simplemente se devuelve los elementos más cercanos a la búsqueda, en este caso, se considera como métrica el producto coseno definido en el punto 2.3.1. La principal limitación de estos sistemas es la escalabilidad con esta aproximación simple, requiere el cálculo y almacenamiento de la distancia entre todas las posibles parejas de datos de la base de datos. Tanto el cómputo como la memoria requerida crecen mediante la norma $\mathcal{O}(n)$, totalmente insostenible para BBDD masivas. Para solucionarlo, se recurren a búsquedas subóptimas, normalmente basadas en algoritmos de agrupación o grafos.

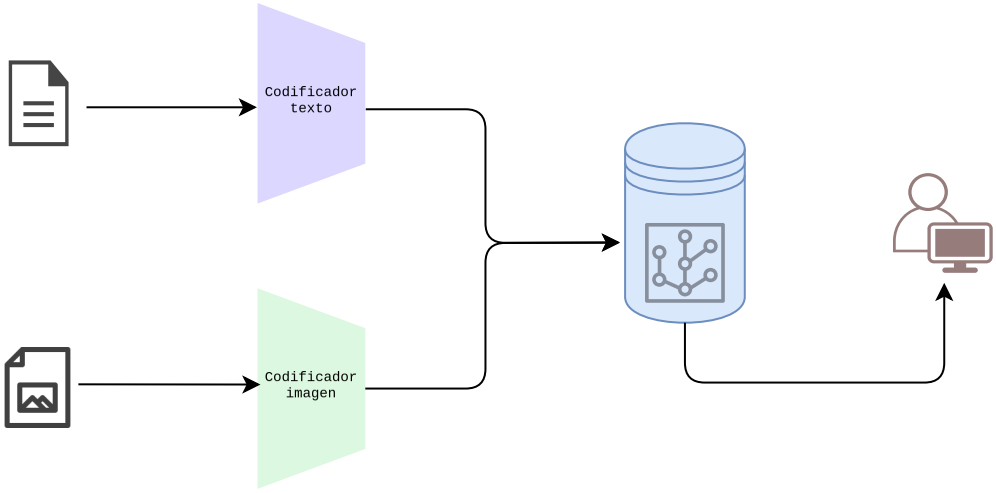

Figura 2.5: Diagrama de un sistema de búsqueda vectorial *CLIP*.

Capítulo 3

El desajuste multimodal

3.1. La brecha intermodal

Dado que por defecto se opera en un espacio altamente dimensional (768 para la versión utilizada de *CLIP*), la visualización de los vectores no se puede realizar directamente. Con el fin de aumentar la interpretabilidad en problemas con vectores con un número de dimensiones elevado, a lo largo de los años se han desarrollado una multitud de técnicas de proyección. La finalidad de este tipo de técnicas es la representación en un espacio con una dimensionalidad menor, pero preservando cualidades o atributos de la distribución en el espacio original, de forma que permitan la interpretación. Las técnicas más populares son el análisis de componentes principales o *PCA* en el ámbito lineal y si se consideran proyecciones no lineales, las técnicas más populares son *t-SNE* [17] y *UMAP* [18].

Al realizar la proyección a un espacio bidimensional mediante todas estas técnicas, se tiene que existe una brecha entre la posición de los vectores de una modalidad con respecto a los de la otra. Este fenómeno, tal y como se puede apreciar en la figura 3.1 persiste en diversas arquitecturas o modelos entrenados mediante el entrenamiento contrastivo autosupervisado.

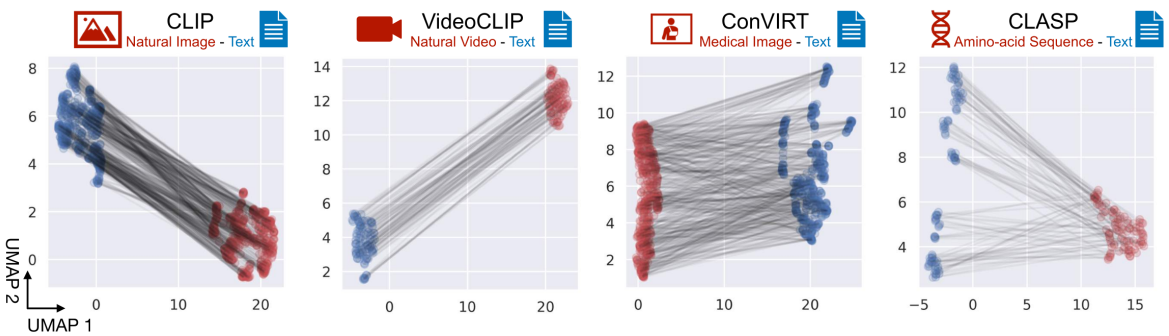

Figura 3.1: Proyección mediante *UMAP* de los vectores densos intermodales para diversos modelos representación vectorial.[1]

El análisis realizado por Liang, W. et al. [19] interpreta la brecha intermodal como la distribución de los vectores densos de cada modalidad en un cono de la hiperesfera. A modo ilustrativo, se muestra en la figura 3.2 el efecto cono ilustrado en una esfera, cabe destacar que los vectores únicamente se encuentran en la corteza esférica debido a que se encuentran normalizados, además se trata de una gran simplificación debido a la incapacidad del ser humano a comprender espacios de mayor dimensión que tres.

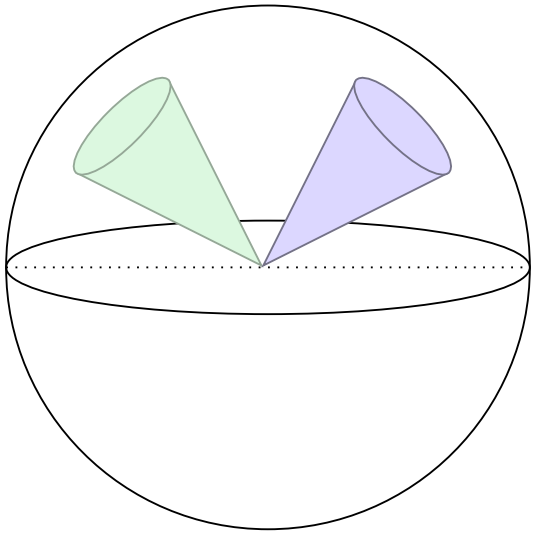

Figura 3.2: Ilustración del efecto cono.

A continuación, se resumen las tres causas de la existencia de la brecha intermodal expuestas por Liang, W. et al. [19].

- Sesgo en las arquitecturas: adjudica a las propias arquitecturas la preferencia de proyección a un cono del espacio global de representación. Para ello, se obtienen los vectores densos de 5.000 imágnenes del conjunto de datos *MSCOCO* con 3 modelos distintos (ResNet, ViT y Transformer de texto), posteriormente se calculan las medias de los productos coseno (0.56, 0.47, 0.51) respectivamente y el mínimo valor de este (0.23, 0.05, 0.01). Esto indica que la distribución de los vectores se encuentra en un cono y no a lo largo de toda la corteza.

- Cada inicialización aleatoria genera su propio cono: se muestrean 25 inicializaciones aleatorias para diversas arquitecturas, se obtienen los vectores resultantes de cada una y se proyectan mediante *UMAP*. Se observa en la figura 3.3 que cada una de las inicializaciones aleatorias proyecta los vectores en un cono distinto de la hiperesfera, además este fenómeno está presente en todas las arquitecturas analizadas tanto texto como imagen.

[1]Fuente: Liang, W. et al. [19].

– El objetivo final del entrenamiento contrastivo mantiene la brecha intermodal: para demostrar este lema, se entrena *CLIP* de forma que se compensa la brecha intermodal mediante el proceso descrito en B.1, se observa que pese a la compensación se vuelve a generar la brecha y de hecho, tiende al valor que presentaba esta originalmente.

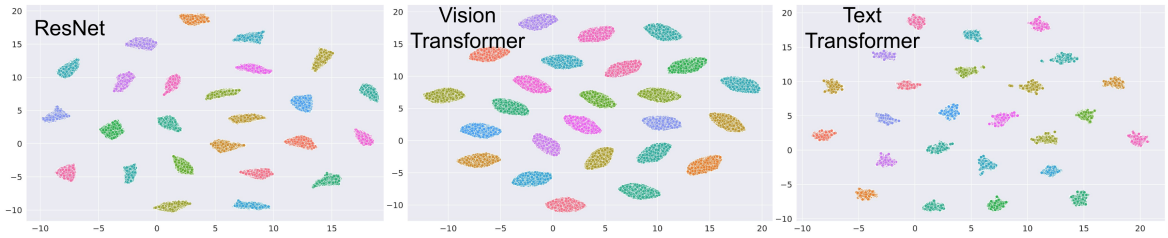

Figura 3.3: Visualización *UMAP* de vectores obtenidos mediante inicializaciones aleatorias.[2]

3.2. Métricas: desalineamiento y la agrupación

Con el objetivo de cuantificar la brecha intermodal, se propone como métrica la distancia entre la media de los vectores de cada modalidad. Considerando $x_i, y_i \in \mathbb{R}^d$ como los vectores correspondientes a cada modalidad, la BRECHA INTERMODAL cuantifica la distancia entre la media de los vectores asociados a cada una de las modalidades, véase la ecuación 3.1. Volviendo al ejemplo de los conos en la hiperesfera, sería análogo a considerar el centro de los conos, no tiene en cuenta la elongación o anchura ocupada en la superficie.

$$\text{BRECHA INTERMODAL} \triangleq \|\mathbb{E}[x_i] - \mathbb{E}[y_i]\|_2 \tag{3.1}$$

Las métricas que se proponen continuación se describen en el trabajo presentado por *Wang, T.* e *Isola, P.* en [20], en él se presenta un análisis de las propiedades de alineamiento y uniformidad en un proceso de optimización mediante un entrenamiento contrastivo. Se hace referencia al alineamiento y la uniformidad, de forma que se consideran como funciones de pérdidas, por lo que se desean minimizar. En este trabajo se hará referencia a estas métricas como DESALINEAMIENTO y AGRUPAMIENTO, de forma que se facilite la comprensión de los resultados.

El desalineamiento se define como el error cuadrático medio entre los vectores correspondientes a un mismo ejemplo, pero entre ambas modalidades, su expresión queda detallada en la ecuación 3.2. Este error se minimiza cuando ambos codificadores

[2]Fuente: Liang, W. et al. [19].

proyectan al mismo punto de la hiperesfera tanto la descripción como la imagen correspondiente a esta.

$$\text{Desalineamiento} \triangleq \mathbb{E}[\|x_i - y_i\|_2^2] \tag{3.2}$$

La medida del agrupamiento o de la uniformidad, como se presenta en [20], no es una tarea trivial. En primer lugar, se exige que la métrica de agrupamiento sea asintóticamente correcta, es decir, que el valor esperado sea 0 cuando se trata de distribución uniforme y además, se requiere que sea razonable con un número finito de puntos. Por lo tanto, se consideran las funciones base radiales (RBF), en concreto potencial gaussiano $G_t : \ S^d \times S^d \to \mathbb{R}_+$ con $G_t(u, v) \triangleq e^{-t\|u-v\|_2^2} = e^{2t \cdot u^T v - 2t}, \ t > 0$, esta función presenta una íntima relación con la distribución óptima de puntos en una esfera [21]. Dado que en [20] se propone hacer uso de esta métrica como una función de pérdidas, se le aplica una transformación logarítmica al valor esperado del potencial gaussiano. El agrupamiento, véase la ecuación 3.3, mide la concentración en la distribución de las diferencias entre vectores pertenecientes a la misma modalidad, se realiza este cálculo en ambas modalidades y se promedia.

$$\text{Agrupamiento} \triangleq \frac{1}{2}\log(\mathbb{E}[e^{-2\|x_i - x_j\|_2^2}]) + \frac{1}{2}\log(\mathbb{E}[e^{-2\|y_i - y_j\|_2^2}]) \tag{3.3}$$

3.3. Técnicas de optimización del entrenamiento

Tal y como se ha comentado con anterioridad, el entrenamiento de este tipo de modelos es intensivo en cuanto al uso de memoria se refiere. Los equipos usados por *OPEN-AI* para el entrenamiento original constan de 256 tarjetas gráficas Nvidia V100, cada una de 32 GB. Nuestro equipo es un poco más modesto, constando en su totalidad de una única tarjeta gráfica de 24 GB. De esta forma es evidente la necesidad de optimizar este proceso de entrenamiento. Para agravar la situación, se debe de tener en cuenta que en el entrenamiento contrastivo, el tamaño de lote juega un papel crucial. Dado que las etiquetas o la supervisión se obtiene entre los propios ejemplos, un mayor tamaño de lote permite una mayor extracción de la información de los ejemplos. A continuación se detallan tres técnicas con el fin de optimizar al máximo los recursos disponibles, en concreto la limitación de memoria.

3.3.1. Puntos de control de gradientes

Una técnica ampliamente utilizada para reducir memoria durante el entrenamiento de modelos consta en descartar gradientes para liberar memoria y calcularlos en el momento. A la hora de entrenar un modelo de redes neuronales profundas

mediante *backpropagation* o propagación hacia atrás de los gradientes, se computa un grafo que posteriormente se diferencia. Este grafo contiene la información sobre las transformaciones aplicadas a los datos y, por lo tanto, mediante autodiferenciación se puede obtener el gradiente a la salida con respecto a cualquier nodo. Para computar el gradiente con respecto a cada nodo, se requiere almacenar los valores de los nodos intermedios. Mediante el uso de puntos de control de gradientes, se descartan estos valores en todos los nodos no designados puntos de control. De esta forma, se consigue liberar memoria, pero sacrificando un tiempo de ejecución mayor, dado que ahora se deben de calcular los valores de los gradientes en los nodos que se han descartado durante la pasada hacia atrás.

Adicionalmente, se ha estudiado otra técnica común para entrenar con mayor memoria de la que sería posible, llamado acumulación de gradientes. Se basa en calcular de forma independiente los gradientes de distintos lotes, almacenarlos y hacer la optimización con el conjunto de los gradientes. De esta forma, teóricamente se permite una convergencia más uniforme y libre de ruido propia de un tamaño de tirada mayor. Esta técnica no es conveniente para el aprendizaje contrastivo, dado que este se basa en la relación entre los distintos ejemplos de una tirada. Esta es una de las grandes limitaciones del aprendizaje contrastivo, su gran demanda de memoria durante el entrenamiento y la necesidad de un tamaño de lote elevado.

3.3.2. Entrenamiento de las últimas capas

Una aproximación común para realizar un ajuste fino en los modelos grandes es el entrenamiento selectivo de las últimas capas. De esta forma se congelan los pesos de todas las capas con excepción de algunas próximas a la salida del modelo. Dado que el problema de la brecha intermodal es un problema de representación, son estas últimas capas las que mayor influencia tienen en este problema. Normalmente, las arquitecturas de redes neuronales profundas presentan un sesgo de diseño, de forma que la información se organiza de forma jerárquica conforme se atraviesan las capas. De esta forma, por ejemplo, para el caso de la imagen, se ha demostrado que las primeras capas responden a la detección de patrones y texturas, mientras que las últimas realizan tareas de representación de la información. De esta forma, se puede interpretar como un proceso de destilación de la información mientras se va atravesando la red.

En nuestro caso, se ha probado al entrenamiento de la red mediante este tipo de técnicas, pero al final se ha optado por la descomposición de bajo rango. Entrenado únicamente las dos últimas capas de ambos modelos y mediante el uso de puntos de control de gradientes, se logra tener un tamaño de lote de 128 ejemplos. Si bien se obtiene un ahorro considerable de memoria, se concluye que los resultados son

altamente dependientes en cuanto al número de capas congeladas. Además de esto, los resultados obtenidos en *MS-COCO* empeoran en la gran parte de los casos con respecto al punto de partida. Esto se debe a que *CLIP* se encuentra en un máximo local desde un punto de vista de la optimización. Este punto es altamente sensible a los parámetros de entrenamiento, por lo que ante la imposibilidad de optimizar todo el conjunto de pesos, los resultados que se obtienen empeoran. Desde un punto de vista forma, se puede interpretar como al congelar gran parte del modelo, se tiene que el subconjunto de tareas que presentan solución se ha reducido considerablemente. De hecho, es probable que la nueva tarea no presente una solución de forma teórica, partiendo de aquellas representaciones intermedias y teniendo en cuenta el número reducido de parámetros entrenables.

3.3.3. Adaptación de bajo rango mediante descomposición de pesos (*DoRA*)

Gracias a la creciente popularización en los últimos años de los modelos de lenguaje de gran tamaño, se ha desarrollado diversas técnicas que permiten el ajuste fino de estos sin la necesidad de entrenar por completo todo el modelo. Una de las técnicas más populares, *LoRA* fue introducida por Hu, E. et al. [22] en 2021, se presenta la adaptación de bajo rango como solución a los requisitos exuberantes de memoria y recursos requeridos para entrenar el modelo por completo. La adaptación de bajo rango consiste en congelar cada una de las capas de los pesos correspondientes al modelo y entrenar, de forma paralela, matrices de descomposición de rango. Se puede observar en la figura 3.4 como estas matrices proyectan a una dimensión inferior, adicionalmente, contienen un número de parámetros mucho menor que cada capa de pesos. De esta forma se combina a la salida de los pesos el resultado del modelo original y el resultado de la descomposición y recomposición de bajo rango.

De forma semejante, *DoRA* [23] se basa en la descomposición de bajo rango, pero además se realiza una descomposición adicional del vector en magnitud y fase. Se ha demostrado empíricamente que se mejoran los resultados mediante el entrenamiento separado del vector de magnitudes y la matriz de fases. En otras palabras, se aplica el mismo procedimiento que *LoRA* a la matriz de fases, y las magnitudes se entrenan directamente mediante el optimizador elegido. En la figura 3.4, se observa de forma visual la descomposición adicional en fase y magnitud.

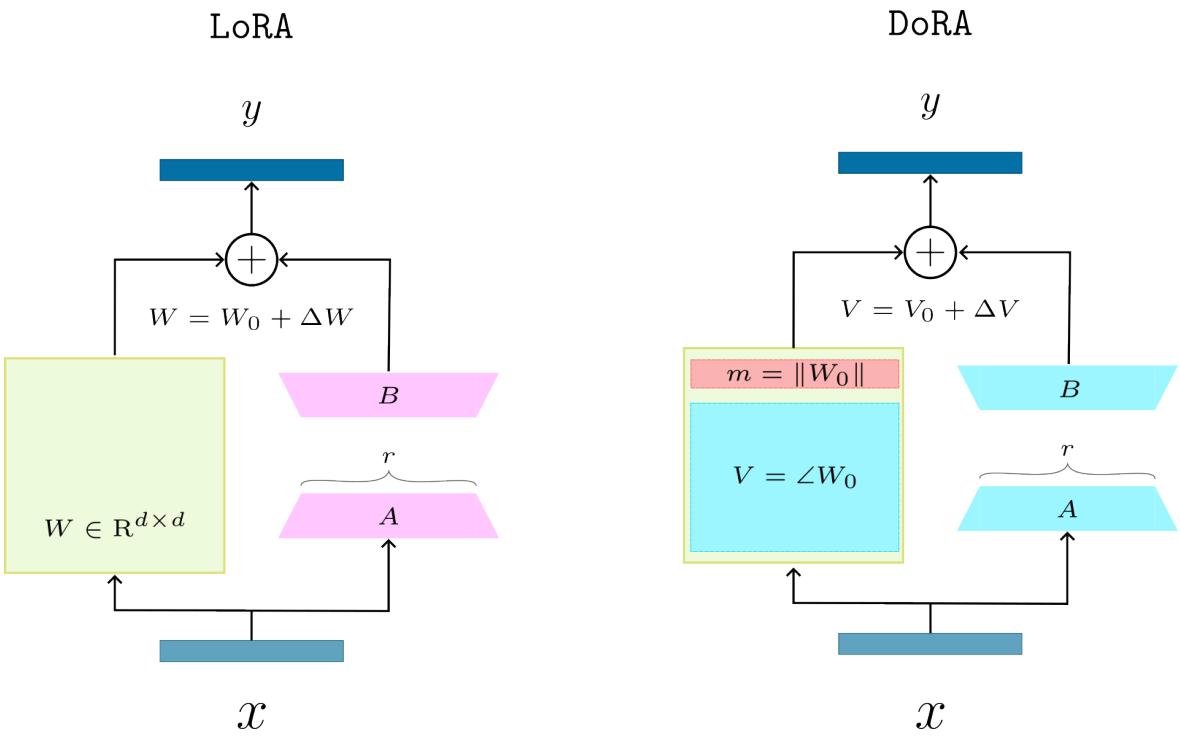

Figura 3.4: Esquema de los métodos de descomposición de bajo rango, *DoRA* y *LoRA*.

Capítulo 4

Bases de datos

4.1. *Microsoft COCO: common objects in text*

COCO (*Common Objects in Context*) es una colección de datos para tareas de detección, segmentación y descripción de imágenes con el fin de mejorar los sistemas para esas tareas. Fue desarrollado por Microsoft para superar las limitaciones existentes en conjuntos de datos de detección de clases en los que el número de clases es reducido y además no se ofrece información adicional de contexto. De esta forma, en *MS-COCO* se tiene tanto una segmentación multiclase como diversas descripciones de la escena. En el caso en cuestión, se hace uso de las descripciones como frases de búsqueda y el objetivo es recuperar la imagen que se describe.

La colección total consta de unas 120.000 imágenes para entrenar, en muchos casos con más de una descripción para cada una y 20.000 imágenes para la evaluación. A continuación, en la figura 4.1, se presentan 3 imágenes pertenecientes a *MS-COCO* junto a sus descripciones.

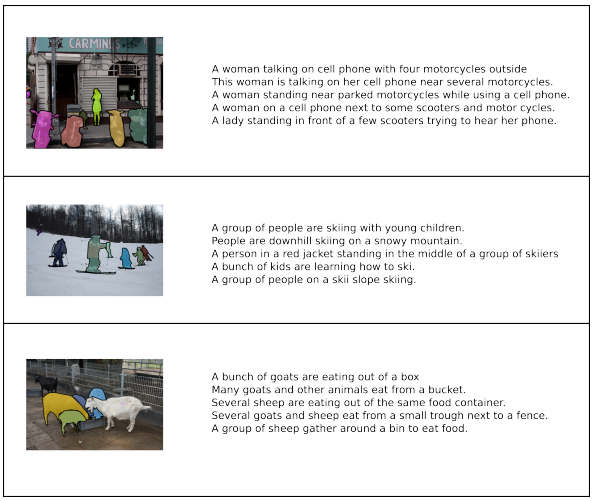

Figura 4.1: Ejemplos de *MS-COCO* junto a sus descripciones.

4.2. Pictogramas *ARASAAC*

Los sistemas aumentativos y alternativos de comunicación (SAAC) son formas de expresión diferentes al lenguaje hablado. Mediante el uso de este tipo de sistemas posibilitan la comunicación de individuos que, debido a una discapacidad o analfabetismo, sería imposible. Entre las causas de uso de un SAAC, se encuentran la discapacidad intelectual, la parálisis cerebral, diversas enfermedades neurológicas como la esclerosis lateral amiotrófica (ELA), el párkinson o simplemente el desconocimiento del idioma, esto último es frecuente en centros de acogida de refugiados.

En este caso, se hace uso de una base de datos de un sistema pictográfico desarrollado por el Portal Aragonés de Comunicación Alternativa y Aumentativa (CAA). Junto al Sistema Pictográfico de Comunicación (SPC), componen los sistemas de CAA con mayor difusión del país.

El sistema se compone de 12260 pictogramas ampliamente etiquetados. A continuación, en la figura 4.2 se muestran un par de pictogramas a modo de ejemplo. Adicionalmente, en el anexo A, se encuentran los archivos `.json` correspondientes a este par de ejemplos junto a una tabla explicativa de todos los campos. En estos, se disponen de las etiquetas de los pictogramas tanto en inglés como en español, esto permitirá la evaluación del modelo en un escenario multilingüe.

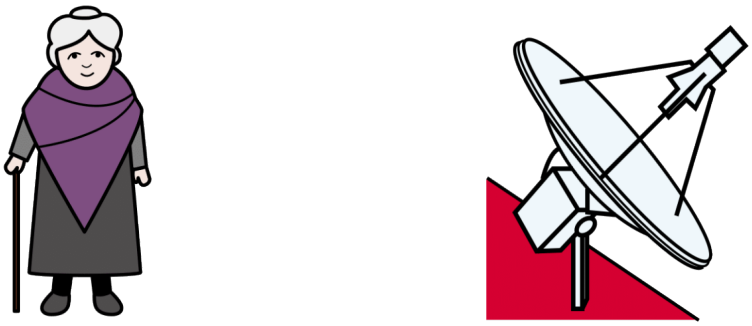

Figura 4.2: Pictogramas de ejemplo correspondientes a **abuela** y **antena**.

Dado que *CLIP* ha sido entrenado con frases o descripciones en lugar de etiquetas, se ha demostrado que se mejora el resultado de la búsqueda considerablemente si se le introduce un prefijo introductorio o *prompt*. Por lo tanto, para las etiquetas en español se busca mediante la frase: `Una imagen de {keyword1}, {keyword2}, ...` De forma análoga, para el caso de las etiquetas en inglés se tiene: `A picture of {keyword1}, {keyword2}, ...`

4.3. Preprocesado de datos

Ambas bases de datos, están formadas mediante imágenes en una carpeta y un archivo `json` con los diversos campos. Como se está trabajando con la librería *PyTorch*, es necesario convertir tanto las imágenes como los textos a un formato compatible con *PyTorch*. Se define una estructura de datos propia de *Pytorch*, llamada `Dataset`. En esta se deben de implementar los métodos de carga inicial de datos y la indexación de un subconjunto de estos. Al hacer uso de esta estructura, mediante el uso de un `DataLoader` permite agilizar la carga de datos, relegando a este las funciones de concurrencia y manejo de lotes.

En la figura 4.3, se muestra el preprocesado de datos de una forma visual. A continuación, se detallan cada una de las partes.

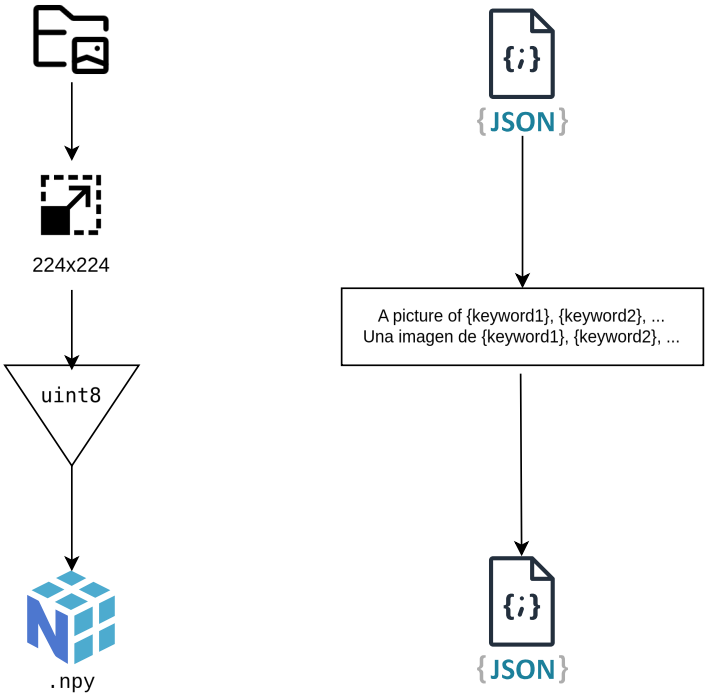

Figura 4.3: Esquema de ingesta de datos.

En cuanto a las imágenes, dado que la lectura a muchos ficheros es más lenta que a un único fichero con todos los datos, se pretende agrupar las imágenes en un único fichero. Se elige el formato `.npy` gracias a la posibilidad de almacenar los datos con un entero de 8 bits (`uint8`) en lugar de un valor en coma flotante de 32 bits (`float32`). Este paso es crucial dado que es la única forma de contener todo el conjunto de datos en memoria RAM. En caso de no haber sido así, se produciría una ralentización considerable, llegándose a pasar la mayor parte del tiempo cargando datos. Además del cambio de formato, es crucial escalar las imágenes al tamaño indicado (224x224). Para no realizar esta tarea con cada imagen entrante al modelo, se escala una única

vez antes de guardarlos en el formato `.npy`.

Por otro lado, el procesado de los textos es mucho más sencillo. En el caso de *MS-COCO*, únicamente es necesario extraer el texto relacionado con la descripción y almacenarlo junto al prefijo en otro archivo `json`. Para el caso de los pictogramas, se deben de extraer de los campos jerárquicos todos los *keywords* asociados al ejemplo en cuestión. Posteriormente, se añade el prefijo y también se almacena en un archivo `json`.

Capítulo 5

Experimentación y análisis de resultados

En el anexo B, se realiza una revisión bibliográfica de los métodos existentes para mitigar la brecha intermodal. A continuación, se resume a grandes rasgos los objetivos de cada método.

- Desplazamiento de vectores: Liang, W. et al. [19] proponen una primera aproximación basada en realizar un desplazamiento a los vectores de cada modalidad. De esta forma, se pretende tener el cono de ambas representaciones alineado.

- Adaptadores para clasificación: se detalla la evolución de los adaptadores de *CLIP* usados para tareas de clasificación. En particular, cabe destacar *Tip-X* [24], donde se tiene en cuenta la inconsistencia del espacio vectorial de imagen.

- *CyCLIP*: se presenta una función de coste con el objetivo de solventar la inconsistencia intramodal. A breves rasgos, se pretende incluir la calibración de los espacios intramodales en la función de coste del model, de esta forma se tienen que todos los productos son consistentes entre sí.

La aproximación que se toma para analizar el efecto de la brecha intermodal es partir del modelo *CLIP* ya entrenado y realizar un ajuste fino mediante cambios en la función de coste. Por ello, se han propuesto un total de 2 funciones de coste, cada una de ellas intentando remediar diversas propiedades de la representación. Ambas funciones de coste, se deben de implementar como términos adicionales a la entropía cruzada descrita en 1.

Inspirados en trabajos como la función de coste propuesta por Kanchana, R. et al. [25], se pretende modificar la función de pérdidas de forma que se optimiza directamente la ortogonalidad entre los vectores densos. De esta forma definimos la siguiente función de coste ortogonal $\mathcal{L}_o$.

$$\mathcal{L}_{o_{intra}} = -\alpha \sum_{i=1}^{N} |\langle x_i, x_i \rangle| + \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} |\langle x_i, x_j \rangle| \tag{5.1}$$

$$\mathcal{L}_{o_{inter}} = -\alpha \sum_{i=1}^{N} |\langle x_i, y_i \rangle| + \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} |\langle x_i, y_j \rangle| \tag{5.2}$$

Siguiendo con la idea de inter/intra modalidad, se define la función de coste ortogonal para los productos inter e intra modales. Se tienen para representaciones latentes de vectores pertenecientes a la misma modalidad ($|\langle x_i, x_j \rangle|$), tanto como para el producto cruzado ($|\langle x_i, y_j \rangle|$). Además, se proporciona un parámetro ajustable α, de forma que modula el efecto en la representación, asociado a las parejas correctas.

Inspirado por la idea de la representación en un cono, se puede interpretar de una forma sencilla que la apertura de este cono es proporcional a la varianza de los productos de las representaciones. Dado que se desea optimizar la separabilidad entre representaciones latentes, una forma de forzar esto es mediante la optimización de la varianza.

Así, se tiene que para representaciones ortogonales entre sí, todos los elementos serán perpendiculares, por lo que el producto coseno asociado a esto es 0. De esta forma se puede ver la minimización de la varianza como una extensión de la función de coste presentada en las ecuaciones 5.2 y 5.1, pero con un momento de segundo orden. Se debe de matizar que optimizando únicamente la varianza en la representación, se llega a una solución óptima en la que todos los vectores se proyectan en el mismo punto. Para evitar esto, se requiere de tener presenta en la función de coste el optimizador de primer orden que condicione la ortogonalidad.

A diferencia del caso anterior, no se va a implementar para los escenarios inter e intra modales, se calcula únicamente para la matriz de productos coseno intermodales. El razonamiento detrás de esta decisión es no agravar el efecto cono ya presente por la arquitectura, como se ha expuesto anteriormente en 3.1.

Para empezar, se define el vector de varianzas S^2, asociada a la matriz de productos coseno intermodales de un lote, mediante el estimador sesgado de varianza. Es necesario calcular de forma separada la varianza asociada a las parejas cruzadas (elementos fuera de la diagonal) con la de las parejas correctas. Esta última es nula para el caso intramodal, su producto coseno es 1 dado que son el mismo vector. De esta forma, se define $S^2_{\not\mathbb{I}}$ como el vector de N elementos que contiene la varianza de los elementos no pertenecientes a la diagonal. Para los elementos de la diagonal se tiene que la varianza es directamente un número, $S^2_{\mathbb{I}}$.

Ambas varianzas se logran mediante el estimador sesgado de varianza, definido de

forma general en la ecuación 5.3. El argumento que minimiza el estimador sesgado y no sesgado de varianza es el mismo en ambos casos. Por tanto, a efectos prácticos, la función de pérdidas de cara a la optimización es equivalente.

$$S^2 = \frac{1}{N}\sum_{i=1}^{N}(d_i - \bar{d}_i)^2 \tag{5.3}$$

$$\mathcal{L}_\sigma = -\beta(\bar{S}_{\mathbb{I}}^2 + S_{\not{\mathbb{I}}}^2) \tag{5.4}$$

De esta forma, se pretende que al optimizar la ortogonalidad entre parejas intermodales y no correspondientes, se llegue a una matriz de productos coseno intermodales con elementos cercanos a cero fuera de la diagonal y próximos a 1 en esta. Se puede ver como la minimización de la varianza en ese caso es ventajosa, de forma que implementa una restricción de uniformidad entre parejas cruzadas. Puede resultar similar a la idea de la consistencia intramodal de *CyCLIP*, pero en este caso se implementa mediante la varianza de los productos coseno.

Tanto la optimización de la ortogonalidad como la de la varianza, se incorporan junto a la entropía cruzada, de forma que se tienen las combinaciones en las ecuaciones 5.5, 5.6 y 5.7. Este paso es crucial, dado que si no se incorpora la entropía cruzada, no se optimiza directamente la tarea de recuperación de información.

$$\mathcal{L}_{CE+o} = \mathcal{L}_{CE} + \mathcal{L}_o \tag{5.5}$$

$$\mathcal{L}_{CE+\sigma} = \mathcal{L}_{CE} + \mathcal{L}_\sigma \tag{5.6}$$

$$\mathcal{L}_{CE+o+\sigma} = \mathcal{L}_{CE} + \mathcal{L}_o + \mathcal{L}_\sigma \tag{5.7}$$

5.1. Visualización de la función de coste para un caso teórico

Una buena forma de entender el objetivo de cada función de coste, resulta por visualizar el valor asociado a una configuración geométrica en el espacio vectorial. Para ello, se propone generar vectores en un espacio tridimensional esférico (norma unidad) de forma que se asemeje al problema original. La distribución que se usa para esta finalidad es la *Power Spherical Distribution* [26]. Se trata de una distribución que permite generar puntos de forma uniforme en una hiperesfera, variando el centro de la distribución, así como un parámetro de concentricidad κ. Se muestra en el anexo C la interfaz junto a los resultados que muestra.

Modificando el valor de la concentricidad se pueden simular el efecto que tendría un cono con una mayor o menor abertura. A continuación, en la figura 5.1 se muestra el efecto de variar el parámetro de concentricidad.

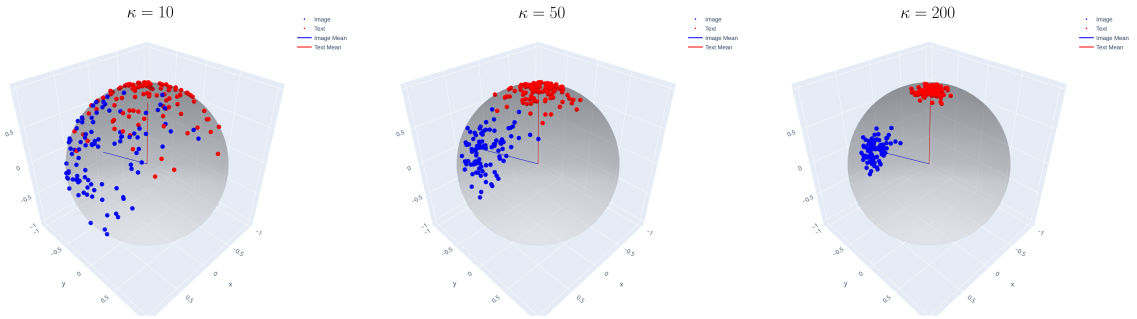

Figura 5.1: Efecto de la concentricidad κ en la distribución.

En cuanto a las métricas de interés, cabe destacar que la medida de desalineamiento no es posible con esta configuración. Debido al proceso generativo, no se realiza una asignación de parejas próximas, de forma que simulen vectores correspondientes al mismo ejemplo. Esto significa que únicamente se miden las estadísticas de las representaciones, cualquier asociación de pares correctos no queda representada en este modelo. Por tanto, cabe esperar que la entropía cruzada sea elevada y sobre todo significativa al modificar la representación.

Con este modelo se pretende obtener una intuición para los parámetros de brecha intermodal y agrupación. En cuanto a la agrupación, se observa que es proporcional al factor de concentricidad encargado de generar la distribución. Además, dado que son distribuciones en un espacio esférico, se tiene que tienen simetría radial (contenida en la corteza esférica), con respecto al punto centra o media. Expresada en la ecuación 3.3, la función de base radial presenta una medida simétrica con respecto al centro de

la distribución de los vectores. Se observa en la tabla 5.1 cómo, en efecto, la agrupación se encuentra totalmente correlada con el valor de concentricidad κ.

κ	Agrupación
200	-0.07
50	-0.265
10	-0.842

Tabla 5.1: Medida de agrupación en función de la concentricidad.

En cuanto a la brecha intermodal, tal y como se describió en la ecuación 3.1, se mide el producto coseno de los centros de los conos generados por cada modalidad. Se propone realizar un barrido para la variable esférica θ, de forma que se aumenta la brecha intermodal. En la figura 5.2, se observan los vectores correspondientes a la media de cada una de las modalidades. Junto con los valores de la tabla 5.2, se observa cómo el valor de la brecha intermodal es proporcional al ángulo entre ambas modalidades.

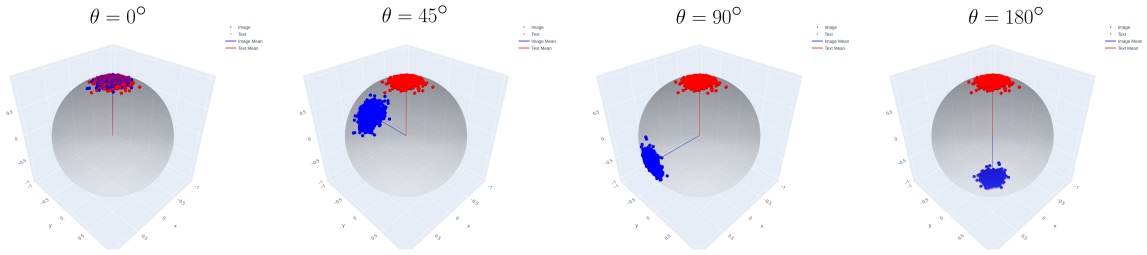

Figura 5.2: Simulación de la brecha intermodal inducida.

θ	Brecha intermodal
0°	≈ 0
45°	0.77
90°	1.40
180°	1.98

Tabla 5.2: Brecha intermodal en función de θ.

Para evaluar el efecto de las funciones de coste, obtienen los valores de las funciones de coste para diferentes configuraciones. Para el caso que nos concierne, únicamente se evalúan los términos adicionales, dado que cómo se ha descrito anteriormente, las parejas no se generan siguiendo un criterio de proximidad. De esta forma, se tiene que la entropía cruzada entre parejas es la equivalente de una elección aleatoria. Por lo tanto, esta es invariante a la configuración geométrica tanto en ángulo como en concentricidad.

En la tabla 5.3 se presentan los resultados en función del ángulo θ entre distribuciones generadoras. Dado que se mantiene un valor de concentricidad constante ($\kappa = 500$), se tiene que las pérdidas intramodales no varían. Se observa cómo, en efecto, la función de pérdidas ortogonal intermodal presenta un mínimo en la configuración ortogonal.

θ	$\mathcal{L}_{o_{intra}}$	$\mathcal{L}_{o_{inter}}$	$\mathcal{L}_{CyCLIP}$	$\mathcal{L}_{\sigma}$
0°	1.94	1.94	3.91	≈0
45°	1.94	1.37	3.92	$-8.13 \cdot 10^{-3}$
90°	1.94	0.00	3.93	$-1.63 \cdot 10^{-2}$
180°	1.94	1.94	3.91	≈0

Tabla 5.3: Valor de las funciones de pérdidas para distintos ángulos θ.

Adicionalmente, en la tabla 5.4, se presentan los valores de las funciones de pérdidas en función de la concentricidad κ. Se ve que *CyCLIP*, tiene un valor de alto para configuraciones muy uniformes, a diferencia del resto de términos analizados. El término asociado a la varianza presenta una magnitud considerable en el menor caso de las concentricidades, queriendo realizar la configuración aún más uniforme. Finalmente, en cuanto a los términos ortogonales, se tiene que la función de pérdidas disminuye inversamente proporcional a la agrupación. Esto se debe al simple hecho de requerir un grado de apertura de cono para que comience a poder ser ortogonal.

κ	$\mathcal{L}_{o_{intra}}$	$\mathcal{L}_{o_{inter}}$	$\mathcal{L}_{CyCLIP}$	$\mathcal{L}_{\sigma}$
10	1.38	1.38	11.28	-0.15
50	1.85	1.85	6.58	$-1.03 \cdot 10^{-2}$
200	1.96	1.96	6.24	≈ 0
500	1.98	1.98	6.21	≈0

Tabla 5.4: Valor de las funciones de pérdidas en función de la concentricidad κ.

5.2. Discusión de resultados

En primer lugar, se estandariza un tamaño de lote de 90 para todos los experimentos. Este parámetro es muy relevante dado que la recuperación de información o *retrieval* (R@k), depende el número de elementos que hay presentes entre los que se pueda elegir. Para todos los experimentos se ha entrenado el adaptador de bajo rango correspondiente al modelo *CLIP* , implementado mediante *DoRA*. La duración del entrenamiento ha sido de 10 épocas, y para todos los casos se ha usado una tasa de aprendizaje de $1 \cdot 10^{-5}$. En el anexo D se muestran los efectos en los histogramas de productos coseno, del entrenamiento con distintas funciones de coste.

5.2.1. *MS-COCO*

Comenzando por *MS-COCO*, existe un problema inmediato: *CLIP* ha sido entrenado con *MS-COCO*. Teniendo en cuenta esto, se espera que se produzca el fenómeno de sobreajuste u *overfitting*, de forma que no se mejora. Se obtiene una gran mejora en el subconjunto de entrenamiento, R@1 del 61.62 % al 99.23 %; sin embargo, en el subconjunto de evaluación el rendimiento mejora en menor medida, pasando del 65.13 % % al 76.82 %. Esta gran diferencia de prestaciones entre subconjuntos se debe a que en el punto en el que se encuentra la red, únicamente puede mejorar su rendimiento, memorizando ejemplos. Estar en esta situación abre la puerta a la comparación entre rendimiento y representación. De esta forma, se podrá determinar si las funciones de coste propuestas actúan sobre la representación con el resultado deseado.

	Brecha inter.	Desal.	Agrup.	Test R@1	Train R@1
Inicial	0.92	1.40	-1.47	65.13	61.62
$\mathcal{L}_{CE}$	0.50	**1.37**	-3.27	**76.82**	98.64
$\mathcal{L}_{CE} + \mathcal{L}_{o_{intra}}$	**0.20**	1.47	**-3.74**	74.35	**99.23**

Tabla 5.5: Parámetros característicos de MS-COCO

Pese a que no una diferencia considerable en el rendimiento, se tiene que la inclusión del término ortogonal lleva a una solución con un valor de brecha intermodal y agrupamiento notablemente menor. En el caso de la brecha intermodal, se consigue reducir esta 0.2 frente a 0.5 para el resto. Este fenómeno únicamente se encuentra presente en el caso de contener la función de coste únicamente los términos ortogonales intramodales. Adicionalmente, el agrupamiento también se minimiza mediante esta configuración, de esta forma se tiene un valor de -3.7 frente a los -3.3 del resto. En cuanto al alineamiento, se tiene el caso contrario, es el que mayor desalineamiento presenta, 1.47 frente a 1.37 del resto. La inclusión tanto del término de minimización

de varianzas como del intermodal de ortogonalidad, rompe tanto con la minimización de la brecha intermodal como con la de la uniformidad.

En primer lugar, en la figura 5.3, se observa el efecto cono descrito en el capítulo 3. Se tiene que las modalidades de imagen y texto individualmente (I y T en el histograma), presentan una media alta. Esto se interpreta como una distribución en cono, pero no quiere decir que se encuentren alineados. Para analizar el alineamiento entre ambas modalidades, se requiere analizar la matriz de productos intermodales (IT). En este caso, se separa esta matriz en los términos correspondientes a las parejas y los cruzados[1].

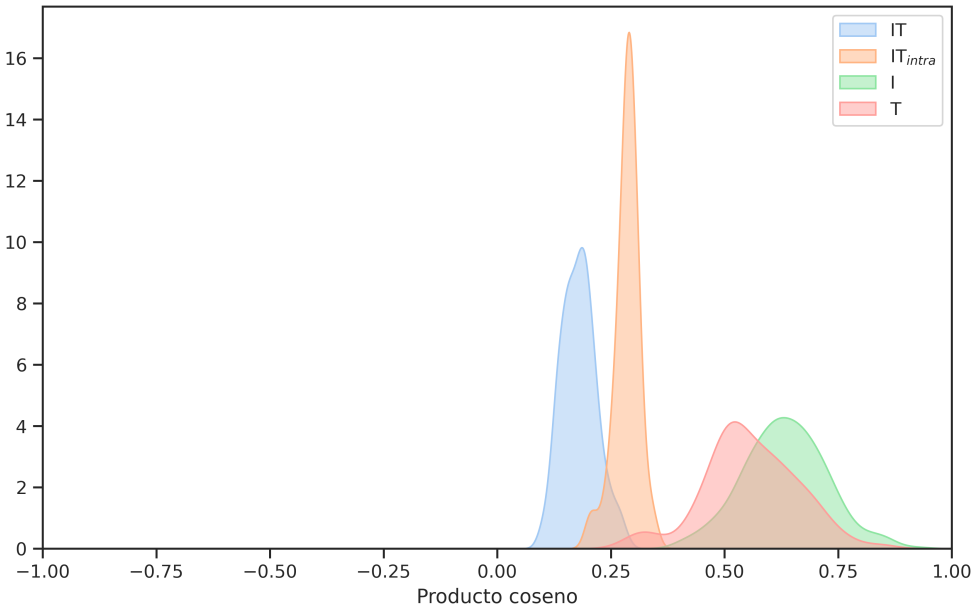

Figura 5.3: Histogramas de productos coseno para *MS-COCO* antes del entrenamiento.

Dado que no se pueden visualizar los vectores directamente y las técnicas de reducción dimensional no aportan la información requerida, se muestra la evolución de los histogramas correspondientes a los productos coseno. Para ello, se separan entre parejas correspondientes (elementos de la diagonal) y el resto. Este proceso se realiza tanto con las matrices de productos coseno intramodales como intermodales.

En la figura 5.4 se observa el efecto que tiene el término ortogonal en la representación vectorial. Se analiza esta representación debido a que el caso de *MS-COCO*, es particular. Con este experimento se pretende demostrar que es posible la optimización directa del objetivo ortogonal. Se ve reflejado cómo el incluir el término ortogonal en la función de pérdidas implica una convergencia mucho más rápida a una solución de este tipo. Pese a que la optimización de la entropía cruzada parece tender

[1]Para el caso de la imagen y del texto, los términos asociados a las parejas correctas tienen un producto coseno de 1, por lo que no se incluyen.

a la solución ortogonal también, se ha visto que se puede reducir la brecha intermodal a la mitad mediante la inclusión de este término, a costa de una ligera pérdida de prestaciones.

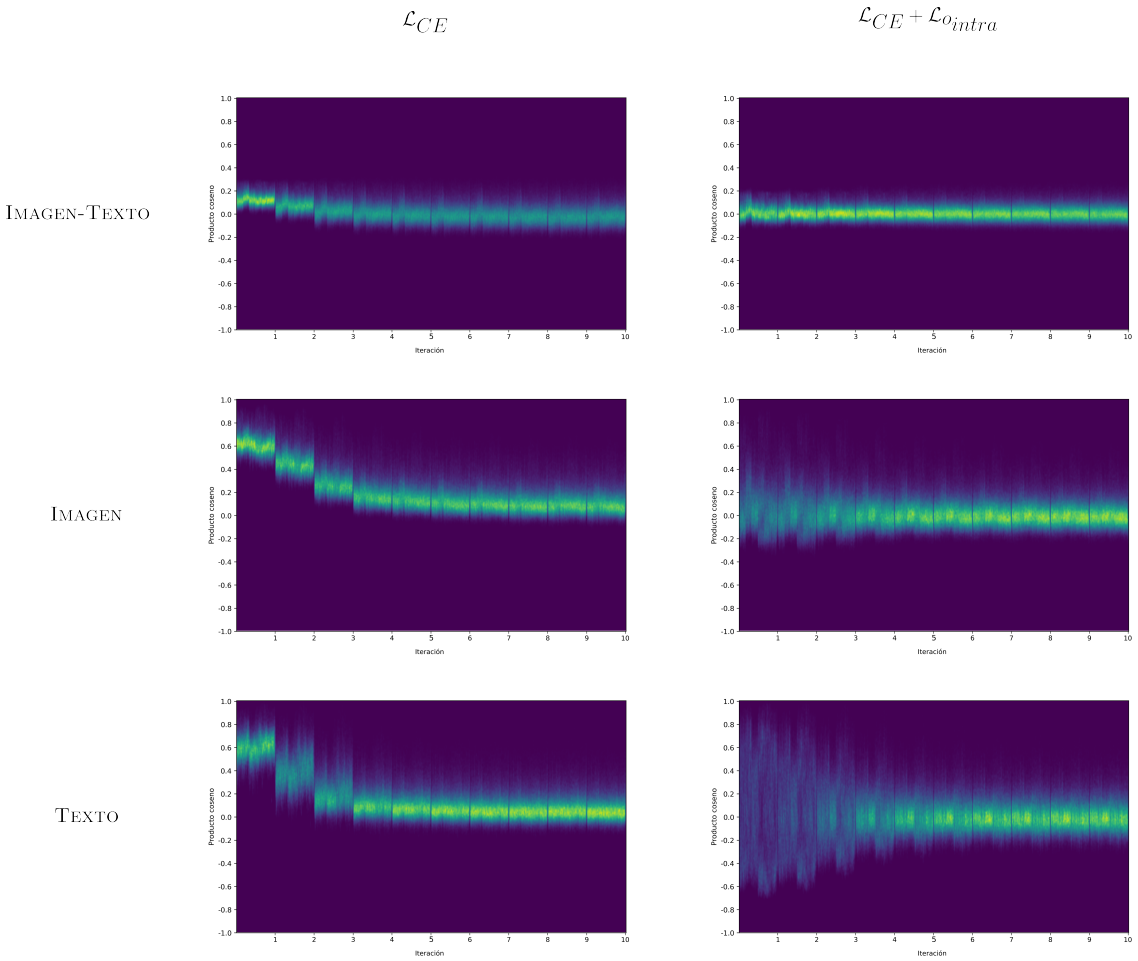

Figura 5.4: Evolución de los histogramas de productos coseno durante el entrenamiento.

5.2.2. Pictogramas ARASAAC en inglés

Habitualmente, para modelos como *CLIP* , se observa que la generalización de conocimiento en imágenes formadas por ilustraciones en lugar de imágenes naturales, es una tarea muy complicada. Brevemente, esto se debe a que es necesario una mayor abstracción para la interpretación del esbozo y de forma que no se puede ayudar de la textura, únicamente de la forma. Dicho esto, se espera que sea posible una mejora significativa en el rendimiento tras el entrenamiento. En la figura 5.5, se observan los histogramas asociados a los productos coseno obtenidos mediante *CLIP* sin entrenar. Se remarca cómo la media de los productos entre vectores de imagen, es significativamente más elevada que en el caso de *MS-COCO*. Presuntamente, se puede deber a que todos los elementos son esbozos, esto corresponde un subconjunto de los datos con los que ha sido entrenado *CLIP* . De esta forma, dado que presenta un grado de similitud implícito, se proyectan en una región cercana en el espacio latente.

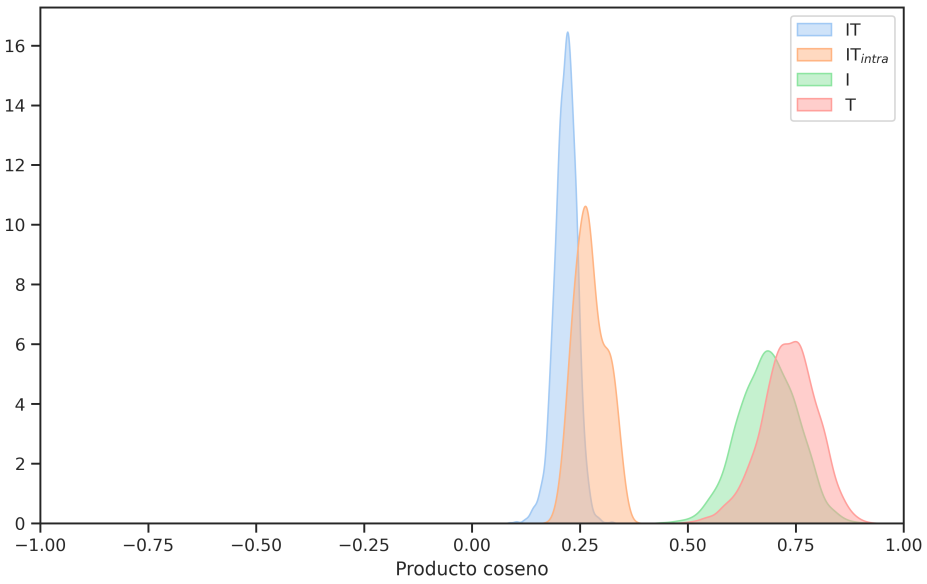

Figura 5.5: Histogramas de productos coseno para los pictogramas en inglés antes del entrenamiento.

La tabla 5.6 muestra los resultados obtenidos al entrenar con las diversas funciones de pérdidas, para los pictogramas en inglés. Para este caso en concreto, se ha dado que la función de pérdidas ortogonal no consigue un entrenamiento favorable en cuanto a rendimiento se refiere. Pese a que se logra un valor de R@1 de 54.5 %, peor que el de partida, el valor de la brecha intermodal desciende hasta 0.32. Para ponerlo en contexto, entrenando únicamente mediante la entropía cruzada, se logra un descenso de la brecha a 0.68. Con el fin de solventar esto, se atribuye el problema a la configuración particular

del espacio vectorial y la ambigüedad resultante de establecer el valor absoluto en la función de coste ortogonal. Si se elimina el valor absoluto se tiene la ecuación mostrada en 5.8, de esta forma se optimiza la antipodalidad. Pese a que la configuración óptima se obtiene en la configuración antipodal, se muestra en el anexo D como sigue tendiendo a ortogonal.

$$\mathcal{L}_{o_{intra}} = -\alpha \sum_{i=1}^{N} \langle x_i, x_i \rangle + \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \langle x_i, x_j \rangle \tag{5.8}$$

Mediante esta nueva función de coste, se observa en la tabla 5.6 un descenso de la brecha intermodal a 0.26. Además, se consigue un valor de *recall* R@1 de 79.6 % frente al 82.2 % , obtenido únicamente mediante la entropía cruzada. En cuanto al desalineamiento, se tiene el mismo caso que con *MS-COCO*, dado que empeora el rendimiento, el desalineamiento aumenta. Finalmente, en cuanto a la agrupación, se tiene que disminuye, por lo tanto, la distribución optimizada resulta más uniforme que la inicial y que la obtenida mediante la entropía cruzada. Finalmente, se tiene el mismo resultado de pérdida considerable de prestaciones, al incluir el término intermodal en la ortogonalización. En este caso, R@1 desciende hasta un mínimo del 18.7 % .

La optimización, concretamente disminución, de la varianza se tiene que por sí sola no presenta ningún efecto. Esto era de esperar, dado que se ha diseñado como un término de segundo orden para función de pérdidas ortogonal. Al combinar la minimización de la varianza junto con el término de la ecuación 5.8, se pierden la gran disminución en brecha intermodal. Parece ser que optimiza una representación similar a la obtenida únicamente mediante el término de entropía cruzada

	Brecha inter.	Desal.	Agrup.	R@1	R@5	R@10
Inicial	1.00	1.40	-1.1	55.8	78.0	84.6
$\mathcal{L}_{CE}$	0.68	1.42	-2.75	**82.2**	95.4	**97.4**
$\mathcal{L}_{CE} + \mathcal{L}_{o_{intra}}$	0.32	1.69	-3.88	54.5	74.6	80.9
$\mathcal{L}_{CE} + \mathcal{L}_{o_{intra}} + \mathcal{L}_{o_{inter}}$	0.32	1.87	**-3.92**	18.7	31.9	40.2
$\mathcal{L}_{CE} + \mathcal{L}_{\sigma}$	0.67	1.40	-2.81	**82.2**	**95.6**	97.3
$\mathcal{L}_{CE} + \mathcal{L}^{*}_{o_{intra}}$	**0.26**	1.56	-3.82	79.6	95.3	96.9
$\mathcal{L}_{CE} + \mathcal{L}^{*}_{o_{intra}} + \mathcal{L}_{\sigma}$	0.64	**1.38**	-2.83	81.8	95.0	97.1

Tabla 5.6: Resultados del entrenamiento de los pictogramas en inglés.

De partida, se tiene que tanto la brecha intermodal como el desalineamiento, son similares para *MS-COCO* y los pictogramas en inglés. Cabe destacar un valor de agrupamiento mayor para los pictogramas en inglés, -1.1 frente a -1.47 en *MS-COCO*. Tal y como se ha expuesto, este valor está directamente correlado con una mayor media en los productos coseno entre imágenes.

5.2.3. Pictogramas ARASAAC en español

Para los pictogramas en español, se tiene un caso muy similar al inglés. Al estar en español y *CLIP* haber sido entrenado principalmente en inglés, se esperan que los resultados sean peores. Así, se mejora la recuperación R@1, desde un 33.2 % por defecto hasta un máximo de 58.4 % .

En cuanto a la distribución de productos coseno, sucede el fenómeno análogo al descrito en el apartado anterior, pero en la modalidad textual. Al tener las etiquetas en español, un subconjunto de los datos de entrenamiento, las representaciones de estas también se encuentran más agrupadas en el espacio latente. Esto se traduce en la figura 5.6 como una mayor media en los productos coseno asociados a los vectores de texto. Adicionalmente, se observa cómo la distribución de productos coseno intermodales de las parejas correctas, se asimila a las incorrectas. Esto se puede interpretar como una pérdida de separabilidad significativa, lo que se traduce en una pérdida de prestaciones.

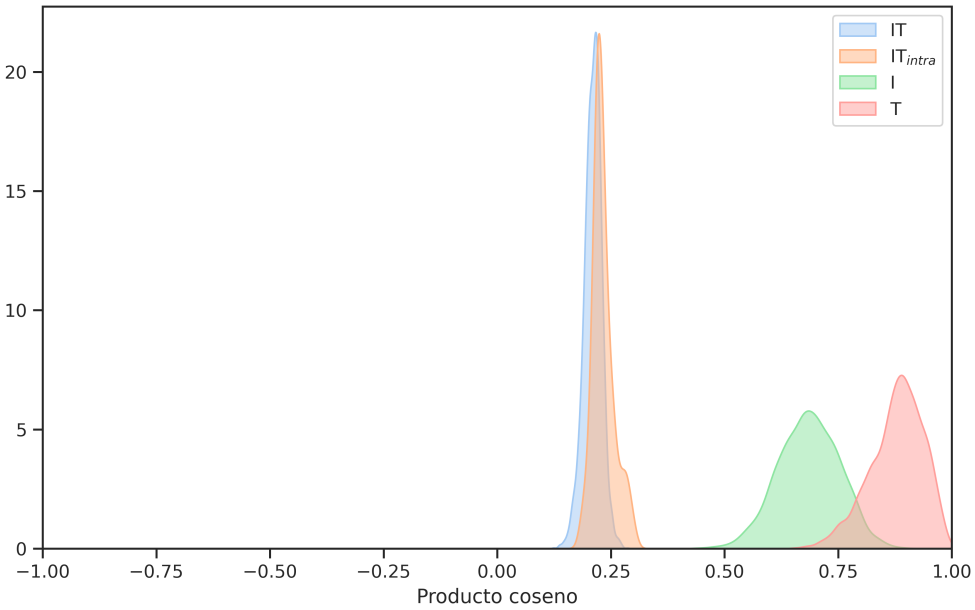

Figura 5.6: Histogramas de productos coseno para los pictogramas en español antes del entrenamiento.

En cuanto a los parámetros de la representación se refiere, se observan diferencias significativas en el desalineamiento y agrupamiento. El desalineamiento, correlado con las prestaciones del sistema, como era de esperar, aumenta. En cuanto al agrupamiento, tal y como se ha descrito, este también se verá incrementado por el efecto cono atribuido al subespacio.

La tabla 5.7 muestra los resultados obtenidos al entrenar con las diversas funciones de pérdidas, para los pictogramas en español. El mejor rendimiento aparece en la tabla cuando se incorpora el término de varianza, logrado un valor de R@1 de 58.4 %. Este resultado se debe a la pura aleatoriedad del proceso de optimización. Realizando varias tiradas, es habitual encontrar una diferencia de un 0.5 %, como se tiene con el término de entropía cruzada únicamente. Se debe de considerar que el resto de valores de R@5 y R@10, R@5 presenta una diferencia del 0.5 % mientras que las prestaciones con R@10 son iguales.

Sorprendentemente, en segundo lugar se encuentra la función de coste antipodal descrita en la ecuación 5.8, logrando un valor de R@1 de 56.3 % . Además del rendimiento, se tiene que logra minimizar la brecha intermodal, hasta 0.27, mucho menor que el 0.52 optimizado mediante la entropía cruzada.

En cuanto al término de varianza, se observa el mismo fenómeno que en el caso anterior. Si se incorpora junto con el término antipodal, la solución a la que se converge es la misma que la ortogonal. De esta forma se consigue una ganancia de un R@1 de 0.9 % a costa de aumentar la brecha intermodal a 0.51.

Pese a que la incorporación de término ortogonal no presenta ventajas frente a la entropía cruzada, si se introduce el término intermodal, se tienen novedades. No solo no se degenera la representación, sino que mejora considerablemente la brecha intermodal, esta se reduce a 0.32. Parece ser que la particularidad de la representación multilingüe permite explorar escenarios que en inglés ya quedan muy definidos. Cabe destacar en todas estas configuraciones la capacidad de minimizar el agrupamiento, partiendo de -0.78 y llegando a -3.88. Este valor mínimo de agrupamiento se consigue con el término antipodal y además con el término ortogonal tanto intramodal como intermodal.

	Brecha inter.	Desal.	Agrup.	R@1	R@5	R@10
Inicial	1.00	1.52	-0.78	33.2	48.5	56.3
$\mathcal{L}_{CE}$	0.52	**1.50**	-3.18	57.1	**77.7**	**84.0**
$\mathcal{L}_{CE} + \mathcal{L}_{o_{intra}}$	0.57	1.69	-3.31	54.6	74.5	81.0
$\mathcal{L}_{CE} + \mathcal{L}_{o_{intra}} + \mathcal{L}_{o_{inter}}$	0.32	1.91	**-3.88**	54.0	74.5	80.0
$\mathcal{L}_{CE} + \mathcal{L}_{\sigma}$	0.50	1.50	-3.23	**58.4**	77.2	**84.0**
$\mathcal{L}_{CE} + \mathcal{L}^{*}_{o_{intra}}$	**0.27**	1.70	**-3.88**	56.3	76.9	82.6
$\mathcal{L}_{CE} + \mathcal{L}^{*}_{o_{intra}} + \mathcal{L}_{\sigma}$	0.51	**1.50**	-3.14	57.2	77.5	83.6

Tabla 5.7: Resultados del entrenamiento de los pictogramas en español.

Capítulo 6

Conclusiones y líneas futuras

6.1. Conclusiones

Las conclusiones que se presentan a continuación son resultados de una extensa revisión bibliográfica junto con una interpretación de los resultados. Pese a que no se ha mejorado el rendimiento añadiendo términos a la función de coste, se pueden extraer conclusiones sobre la representación optimizada. De hecho, los métodos propuestos consiguen manipular efectivamente la representación vectorial. Adicionalmente, se ha demostrado la viabilidad de la optimización mediante una aproximación de bajo rango, como *DoRA*. Una novedad es el análisis de las propiedades de la representación vectorial para un caso multilingüe. Tal y como se expuso en 5.2, existen diferencias en la representación vectorial entre idiomas, principalmente asociadas al entrenamiento por defecto de *CLIP* .

Se observa la existencia de un sesgo en cuanto a brecha intermodal se refiere, en la representación vectorial. Esto implica que no es necesario mitigar la brecha intermodal por completo para que la representación presente su configuración óptima. En cuanto al agrupamiento, se observa que disminuye gracias a la incorporación del término ortogonal. Pese a que parezca una cualidad deseable de la representación, se tiene el mismo resultado que con la brecha intermodal. No se requiere de una representación completamente uniforme para maximizar el rendimiento del sistema. El desalineamiento, tal y como se introdujo, representa el error cuadrático medio entre los vectores intermodales. A groso, modo, el argumento que minimiza la función de coste es equivalente para la entropía cruzada y MSE bajo las condiciones de tener distribuciones gaussianas. Se observa un grado de correlación entre el rendimiento y el desalineamiento, resultante de ser los casos con entropía cruzada los que mejor rinden.

6.1.1. Los espacios altamente dimensionales no se comportan de forma intuitiva

Pese a que en el apartado 5.1, se presenta una herramienta de visualización de los parámetros para vectores tridimensionales, se tiene que la geometría en hiperespacios euclídeos no es intuitiva. Para ilustrar esta afirmación, se tiene que en un espacio euclídeo de un número elevado de dimensiones, el volumen ocupado por un hipercubo de lado 2, es de órdenes de magnitud mayor que el de la hiperesfera de radio unidad. De hecho, el ratio entre volumen del hipercubo y de la hiperesfera tiende a infinito conforme crece la dimensión. Se proporciona en el anexo E la demostración correspondiente, así como una explicación más detallada de algunas propiedades contra intuitivas en espacios euclídeos altamente dimensionales.

Toda esta problemática se ve agravada por el hecho de tener un espacio vectorial cuyas representaciones son altamente no lineales. Si bien la función de pérdidas puede favorecer ciertas propiedades de la representación, la no linealidad existente, imposibilita la extrapolación lineal de estas. Si una reducción dimensional pierde propiedades, es evidente que condensar toda la representación a un número, no representa adecuadamente la casuística presente.

6.1.2. Una función de coste más separable lleva a una representación menos transferible

En este trabajo se atribuye la disminución en rendimiento a la exigencia de un mayor número de propiedades a la representación vectorial. Al imponer un mayor número de requerimiento sobre este espacio, el subconjunto de posibles soluciones se estrecha. Este hecho complica la optimización, llegando a que no se puedan cumplir en el mismo nivel la asociación entre vectores correctos.

La mayor separabilidad interclase implica una mayor precisión de clasificación, todo ello a coste de una menor variabilidad en la representación intraclase. Esto lleva a tener representaciones de mejor calidad.

Este trabajo afirma los resultados de [27], teniendo en cuenta que la tarea de recuperación de información es muy similar las evaluadas por ellos. Dado que con *CLIP* se valora la generalidad de las representaciones, el empeoramiento visto en estos resultados concuerda con lo expuesto anteriormente. Viendo que optimizado diversas métricas, las diferencias se encuentran en las últimas capas, se tiene que la decisión de entrenar todas las capas mediante una aproximación de bajo rango es más acerada que entrenar únicamente las últimas.

6.2. Líneas futuras

Si bien se ha explorado el efecto de la ortogonalidad como mecanismo de separabilidad y reducción de la brecha intermodal, existe una infinidad de métodos posibles. Se proponen dos líneas principales para trabajos futuros, la primera se basa en el análisis de los codificadores desde un punto de vista de la teoría de la información. Se debería de analizar las propiedades de la representación vectorial teniendo en cuenta que la información mutua entre los datos de distintas modalidades no es la misma. Así se podría incorporar la información contenida por cada una de las modalidades, explicando el efecto de representar varias imágenes mediante una única descripción. Finalmente, queda abierto el análisis de la arquitectura sobre el efecto de la representación. Mediante el uso de técnicas comparación como en [27], junto a diversas arquitecturas, se puede determinar si existe un sesgo en ellas.

Capítulo 7

Bibliografía

[1] Calvin Mooers. The theory of digital handling of non-numerical information and its implications to machine economics. University of Michigan, 1950.

[2] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R.A. Harshman. Indexing by latent semantic analysis. Journal of the American Society for Information Science 41, pages 391–407, 1990.

[3] Zellig S. Harris. Distributional structure. WORD, 10(2-3):146–162, 1954.

[4] KAREN SPARCK JONES. A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation, 28(1):11–21, Jan 1972.

[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, Advances in Neural Information Processing Systems, volume 26. Curran Associates, Inc., 2013.

[6] Veenu Rani, Syed Tufael Nabi, Munish Kumar, Ajay Mittal, and Krishan Kumar. Self-supervised learning: A succinct review. Archives of Computational Methods in Engineering, 30(4):2761–2775, May 2023.

[7] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 1, pages 539–546 vol. 1, 2005.

[8] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. ArXiv, abs/1807.03748, 2018.

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh, editors, Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 1597–1607. PMLR, 13–18 Jul 2020.

[10] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stephane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In Marina Meila and Tong Zhang, editors, Proceedings of the 38th International Conference on Machine Learning, volume 139 of Proceedings of Machine Learning Research, pages 12310–12320. PMLR, 18–24 Jul 2021.

[11] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael G. Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 15619–15629, 2023.

[12] Yuhao Zhang, Hang Jiang, Yasuhide Miura, Christopher D. Manning, and C. Langlotz. Contrastive learning of medical visual representations from paired images and text. In Machine Learning in Health Care, 2020.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In North American Chapter of the Association for Computational Linguistics, 2019.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016.

[15] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, Proceedings of the 38th International Conference on Machine Learning, volume 139 of Proceedings of Machine Learning Research, pages 8748–8763. PMLR, 18–24 Jul 2021.

[16] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[17] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. Journal of Machine Learning Research, 9(86):2579–2605, 2008.

[18] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. Umap: Uniform manifold approximation and projection. Journal of Open Source Software, 3(29):861, 2018.

[19] Weixin Liang, Yuhui Zhang, Yongchan Kwon, Serena Yeung, and James Zou. Mind the gap: Understanding the modality gap in multi-modal contrastive representation learning. In NeurIPS, 2022.

[20] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In Hal Daumé III and Aarti Singh, editors, Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 9929–9939. PMLR, 13–18 Jul 2020.

[21] Henry Cohn and Abhinav Kumar. Universally optimal distribution of points on spheres. Journal of the American Mathematical Society, 20(1):99–148, September 2006.

[22] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685, 2021.

[23] Shih yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. ArXiv, abs/2402.09353, 2024.

[24] Vishaal Udandarao. Understanding and fixing the modality gap in vision-language models. Master's thesis, University of Cambridge, 2022.

[25] Kanchana Ranasinghe, Muzammal Naseer, Munawar Hayat, Salman Hameed Khan, and Fahad Shahbaz Khan. Orthogonal projection loss. 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pages 12313–12323, 2021.

[26] Nicola De Cao and Wilker Aziz. The power spherical distrbution. Proceedings of the 37th International Conference on Machine Learning, INNF+, 2020.

[27] Simon Kornblith, Ting Chen, Honglak Lee, and Mohammad Norouzi. Why do better loss functions lead to less transferable features? In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors,

Advances in Neural Information Processing Systems, volume 34, pages 28648–28662. Curran Associates, Inc., 2021.

[28] Shashank Goel, Hritik Bansal, Sumit Bhatia, Ryan Rossi, Vishwa Vinay, and Aditya Grover. Cyclip: Cyclic contrastive language-image pretraining. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, Advances in Neural Information Processing Systems, volume 35, pages 6704–6719. Curran Associates, Inc., 2022.

[29] Renrui Zhang, Wei Zhang, Rongyao Fang, Peng Gao, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. Tip-adapter: Training-free adaption of clip for few-shot classification. In Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXV, page 493–510, Berlin, Heidelberg, 2022. Springer-Verlag.

[30] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. CoRR, abs/1709.00029, 2017.

[31] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[32] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. Clip-adapter: Better vision-language models with feature adapters. International Journal of Computer Vision, 132(2):581–595, Feb 2024.

[33] Malik Boudiaf, Jérôme Rony, Imtiaz Masud Ziko, Eric Granger, Marco Pedersoli, Pablo Piantanida, and Ismail Ben Ayed. A unifying mutual information view of metric learning: Cross-entropy vs. pairwise losses. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, Computer Vision – ECCV 2020, pages 548–564, Cham, 2020. Springer International Publishing.

Lista de Figuras

1.1. Diagrama de *Gantt* para el desarrollo del proyecto. 3

2.1. Clasificación de los paradigmas de aprendizaje automático. 8
2.2. Diagrama de la arquitectura *ConVIRT*. [1] 11
2.3. Diagrama de la arquitectura de *CLIP*.[2] 12
2.4. Diagrama de clasificación mediante *CLIP*.[3] 15
2.5. Diagrama de un sistema de búsqueda vectorial *CLIP*. 16

3.1. Proyección mediante *UMAP* de los vectores densos intermodales para diversos modelos representación vectorial.[4] 17
3.2. Ilustración del efecto cono. 18
3.3. Visualización *UMAP* de vectores obtenidos mediante inicializaciones aleatorias.[5] 19
3.4. Esquema de los métodos de descomposición de bajo rango, *DoRA* y *LoRA*. 23

4.1. Ejemplos de *MS-COCO* junto a sus descripciones. 24
4.2. Pictogramas de ejemplo correspondientes a **abuela** y **antena**. 25
4.3. Esquema de ingesta de datos. 26

5.1. Efecto de la concentricidad κ en la distribución. 31
5.2. Simulación de la brecha intermodal inducida. 32
5.3. Histogramas de productos coseno para *MS-COCO* antes del entrenamiento. 35
5.4. Evolución de los histogramas de productos coseno durante el entrenamiento. 36
5.5. Histogramas de productos coseno para los pictogramas en inglés antes del entrenamiento. 37
5.6. Histogramas de productos coseno para los pictogramas en español antes del entrenamiento. 39

B.1. Perceptrón multicapa con conexión residual. 57
B.2. Representación de los diversos métodos de adaptación. 58

C.1. Interfaz para la visualización de las funciones de pérdidas con distintas configuraciones espaciales. 62

D.1. Evolución de los histogramas distancias. 64
D.2. Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE}$. 65
D.3. Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE} + \mathcal{L}_{o_{intra}}$. . . . 65
D.4. Evolución de los histogramas distancias. 66
D.5. Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE} + \mathcal{L}_{o_{intra}} + \mathcal{L}_{o_{inter}}$. 67
D.6. Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE} + \mathcal{L}_{\sigma}$. 67
D.7. Evolución de los histogramas distancias. 68
D.8. Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE} + \mathcal{L}^{*}_{o_{intra}}$. . . . 69
D.9. Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE} + \mathcal{L}^{*}_{o_{intra}} + \mathcal{L}_{\sigma}$. 69
D.10. Evolución de los histogramas distancias. 70
D.11. Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE}$. 71
D.12. Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE} + \mathcal{L}_{o_{intra}}$. . . . 71
D.13. Evolución de los histogramas distancias. 72
D.14. Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE} + \mathcal{L}_{o_{intra}} + \mathcal{L}_{o_{inter}}$. 73
D.15. Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE} + \mathcal{L}_{\sigma}$. 73
D.16. Evolución de los histogramas distancias. 74
D.17. Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE} + \mathcal{L}^{*}_{o_{intra}}$. . . . 75
D.18. Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE} + \mathcal{L}^{*}_{o_{intra}} + \mathcal{L}_{\sigma}$. 75

Lista de Tablas

5.1. Medida de agrupación en función de la concentricidad. 32
5.2. Brecha intermodal en función de θ. 32
5.3. Valor de las funciones de pérdidas para distintos ángulos θ. 33
5.4. Valor de las funciones de pérdidas en función de la concentricidad κ. . . 33
5.5. Parámetros característicos de MS-COCO 34
5.6. Resultados del entrenamiento de los pictogramas en inglés. 38
5.7. Resultados del entrenamiento de los pictogramas en español. 40

Anexos

Anexos A

Ejemplo de la base de datos de pictogramas de ARASAAC

Con el fin de ilustrar los campos disponibles en la base de datos, se muestran tanto una tabla como los archivos `json` correspondientes a los pictogramas de **abuela** y **antena**, descritos en la sección 4.2.

Campos de la BBDD de pictogramas ARASAAC	
shematic	▷ Se trata de un diagrama
sex	▷ Indica si el pictograma representa contenido sexual
violence	▷ Indica si el pictograma representa contenido violento
aac	▷ AAC presente
aacColor	▷ Color AAC, si procede
skin	▷ Color de piel, si procede
hair	▷ Color del pelo, si procede
downloads	▷ Número de descargas (inoperativo)
categories	▷ Categías a las que pertenece
synsets	▷ Identificador del conjunto de palabras con misma semántica
tags	▷ Etiquetas asociadas a la palabra
created	▷ Fecha de creación
lastUpdated	▷ Última fecha de actualización
keywords	▷ Subconjunto de palabras clave que desciben el pictograma
keyword	▷ Palabra clave o acción principal representada en el pictograma
type	▷ Identificador de tipo de palabra
meaning	▷ Definición en español
plural	▷ Plural de la palabra clave, si procede
hasLocution	▷ Indica si está disponible la locución

Datos del pictograma de antena.


```
"2253": {
    "schematic": false,
    "sex": false,
    "violence": false,
    "aac": false,
    "aacColor": false,
    "skin": false,
    "hair": false,
    "downloads": 0,
    "categories": [
      "mass media device"
    ],
    "synsets": [
      "03212026-n"
    ],
    "tags": [
      "object",
      "appliance",
      "mass media device",
      "mass media"
    ],
    "_id": 2253,
    "created": "2007-12-12T10:27:32.000Z",
    "lastUpdated": "2020-06-23T14:45:46.217Z",
    "keywords": [
      {
        "keyword": "antena",
        "type": 2,
        "meaning": "Dispositivo de los aparatos emisores o
        ↪  receptores que, con formas muy diversas, sirve para
        ↪  emitir o recibir ondas electromagnéticas.",
        "plural": "antenas",
        "hasLocution": true
      },
      {
        "keyword": "parabólica",
        "type": 2,
        "meaning": " U. t. c. s. f. Se dice de la antena
        ↪  radioeléctrica con forma de parábola, y especialmente
        ↪  de la televisión, que permite captar emisoras
        ↪  situadas a gran distancia.",
        "plural": "parabólicas",
        "hasLocution": true
      }
    ]
  }
```

Datos del pictograma de abuela.


```
"10194": {
    "schematic": false,
    "sex": false,
    "violence": false,
    "aac": false,
    "aacColor": false,
    "skin": true,
    "hair": true,
    "downloads": 0,
    "categories": [
      "elderly",
      "family"
    ],
    "synsets": [
      "10162267-n",
      "10068026-n",
      "01648667-s"
    ],
    "tags": [
      "person",
      "elderly",
      "family"
    ],
    "_id": 10194,
    "created": "2009-11-11T19:39:37.000Z",
    "lastUpdated": "2020-11-26T10:23:10.335Z",
    "keywords": [
      {
        "keyword": "abuela",
        "type": 2,
        "meaning": "f. Respecto de una persona, madre de su padre
        ↪  o de su madre.",
        "plural": "abuelas",
        "hasLocution": true
      },
      {
        "keyword": "yaya",
        "type": 2,
        "meaning": "f. abuela",
        "plural": "yayas",
        "hasLocution": true
      }
    ]
  }
```

Anexos B

Revisión de soluciones

Kornbilth, S., et al. analizan, en [27], el efecto de distintas funciones de coste sobre las representaciones densas obtenidas en un problema de clasificación. A diferencia de este trabajo, la tarea que se evalúa es la clasificación, sin embargo, se pretende hacer uso de las representaciones vectoriales para diversas tareas. Para evaluar la calidad de estas representaciones, se propone el uso de un clasificador lineal con estas o mediante el algoritmo de agrupación *K-NN*. De esta forma, se tienen funciones de coste, un mejor resultado de precisión que la entropía cruzada. Sin embargo, cuando se hace uso de las representaciones vectoriales, siempre se obtiene el mejor rendimiento con la entropía cruzada. Adicionalmente, analizan el efecto de cada capa en las distintas representaciones, para ello se utiliza un método conocido como *Linear Centered Kernel Alignment* (CKA). Se observa que la mayoría de representaciones intermedias son similares para las distintas funciones de coste, a excepción de las obtenidas en las dos últimas capas. De ahí que se pueda interpretar como un sobreajustse en la representación final.

Boudiaf, M. et al. [33] presentan un análisis teórico destinado a relacionar la entropía cruzada con diversas funciones de coste, entre ellas la optimización directa de los productos coseno. En primer lugar, establecen una relación entre algunas funciones de coste y el punto de vista generativo de la expresión de información mutua. Se demuestra que minimizar la entropía cruzada, es equivalente a maximizar la información mutua entre las representaciones y las etiquetas, desde un punto de vista discriminativo. De esta forma, se concluye que pese a existir funciones de pérdidas con mejores propiedades, desde un punto de vista de la optimización, empíricamente se demuestra que la entropía cruzada consigue resultados de estado del arte.

B.1. Desplazamiento de vectores

Una primera aproximación propuesta por Liang, W. et al. [19] para mitigar la brecha intermodal, se basa en calcular el vector entre la media de cada uno de los conos y trasladar los vectores en esa dirección. Se define el vector diferencia como $\vec{\Delta} = \mathbb{E}[x_i] - \mathbb{E}[y_i]$, de esta forma se pueden desplazar los vectores correspondientes a cada una de las modalidades: $\tilde{x}_i = \|x_i - \lambda\vec{\Delta}\|_2^2$ y $\tilde{y}_i = \|y_i - \lambda\vec{\Delta}\|_2^2$.

Para evaluar el efecto de variar la brecha intermodal mediante la aproximación detallada anteriormente, se hace uso del esquema de clasificación descrito en la sección 2.3.3 en los siguientes conjuntos de evaluación. En primer lugar, para tareas de clasificación *CIFAR10* [31] y *CIFAR100* [31], para una clasificación específica como imágenes satélite *EuroSAT* [30]. Los resultados presentados por Liang, W. et al. [19] son cuestionables, si bien se presentan un p-valor del orden de 10^{-6} para las pruebas de *CIFAR10* y *EuroSat* y del orden de 10^{-3} para *CIFAR100*[1], en cualquier caso la diferencia la tasa de acierto en la clasificación no sufre cambios mayores al 1 %. Por ende, se concluye que este método de modificación de la brecha intermodal no presenta ventajas significativas, aún más teniendo en cuenta que se trata de una traslación lineal en un espacio vectorial altamente no lineal, debido a los modelos utilizados para codificar los datos.

B.2. La consistencia intermodal e intramodal

B.2.1. Adaptadores para clasificación

La representación vectorial obtenida mediante *CLIP*, gracias al entrenamiento generalista, consigue capturar una gran variedad de características de los datos, de ahí su facilidad de uso para problemas de clasificación. Normalmente en estos casos se parte de un modelo generalista ya entrenado y se realiza un ajuste fino con datos pertenecientes únicamente de la tarea que se pretende optimizar. Este proceso requiere del entrenamiento de todos los pesos del modelo, pese a que *CLIP* no se considera un modelo grande mediante los estándares actuales, su entrenamiento requiere de un sistema con una gran memoria por el tamaño inevitable de las imágenes. Adicionalmente, el ajuste fino de un modelo requiere de una búsqueda exhaustiva de parámetros, además no se garantiza obtener una gran mejora. Esto se debe a que se tiene un modelo altamente sobreparametrizado para una tarea, lo que conlleva una convergencia más lenta, si esta existe.

[1] Nótese la disminución del p-valor para la tarea de clasificación de *CIFAR100*, esta presenta 100 clases frente a las 10 de los anteriores.

Los adaptadores clásicos se basan en esquemas de proyección tanto lineal como no lineal, de esta forma se aprovecha directamente la representación latente en lugar de entrenar desde cero el modelo completo. Una forma de implementar este tipo de sistemas es mediante el uso de un perceptrón multicapa, en los ejemplos descritos a continuación se utilizan concretamente 2 capas. La entrada y salida, presentan las mismas dimensiones que el vector de la representación latente. Sin embargo, la capa intermedia se elige con una dimensión menor[2], de forma que se exige una destilación de la información en el entrenamiento. Adicionalmente, para mitigar el sobreajuste, se incorpora una conexión residual en cada uno de los perceptrones multicapa. Entonces, se combina suavemente el aprendizaje de *CLIP* con el entrenado en la red, véase la figura B.1.

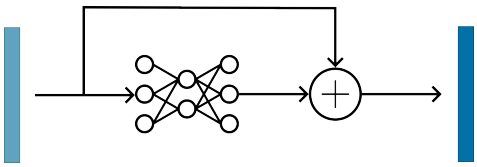

Figura B.1: Perceptrón multicapa con conexión residual.

Mediante el uso de un adaptador en cada modalidad, Gao, P. et al. [32] proponen el método *CLIP-Adapter*, de esta manera se entrenan los adaptadores mediante descenso de gradiente con un banco de imágenes I_c. Se logran mejorar los resultados respecto a *CLIP* preentrenado en una diversidad de pruebas de clasificación de imagen, para un mayor detalle y comparativa entre los diversos métodos, véase la figura B.2.

Zhang, R. et al. [29] proponen el uso de un adaptador, de forma que no requiere entrenamiento. La salida a la red del adaptador descrito anteriormente presenta la siguiente ecuación: $\alpha f(x^T W_1 + b_1)W_2 + b_2 + x$, los parámetros entrenables son W_1, W_2, b_1 y b_2, f es una función de activación no lineal, $f = \mathsf{ReLU}$. En lugar de entrenar estos parámetros, Zhang, R. et al. [29] proponen las siguientes modificaciones: los pesos de la primera capa se sustituyen por la matriz transpuesta de la caché, de esta forma se tienen las similitudes entre los ejemplos proporcionados y la imagen a clasificar. Posteriormente, la función de activación se sustituye por una función exponencial de la forma $g(x) = e^{\beta(1-x)}$. Teniendo en cuenta que las similitudes no pueden tener un valor mayor a 1, esta función exponencial convierte valores negativos en positivos y mediante el parámetro β permite modular la brusquedad del valor. Finalmente, este resultado se multiplica por la matriz de unos y ceros, de forma que cada componente del vector resultante contiene la suma de todos los productos correspondientes a una misma clase. De forma similar que en *CLIP-Adapter*, este valor se ve escalado con otro

[2]La dimensión oculta es de 256 en los sistemas propuestos.

parámetro (α) sin embargo, la conexión residual no es del vector, sino de los productos con los puntajes entre la imagen de entrada y las distintas clases.

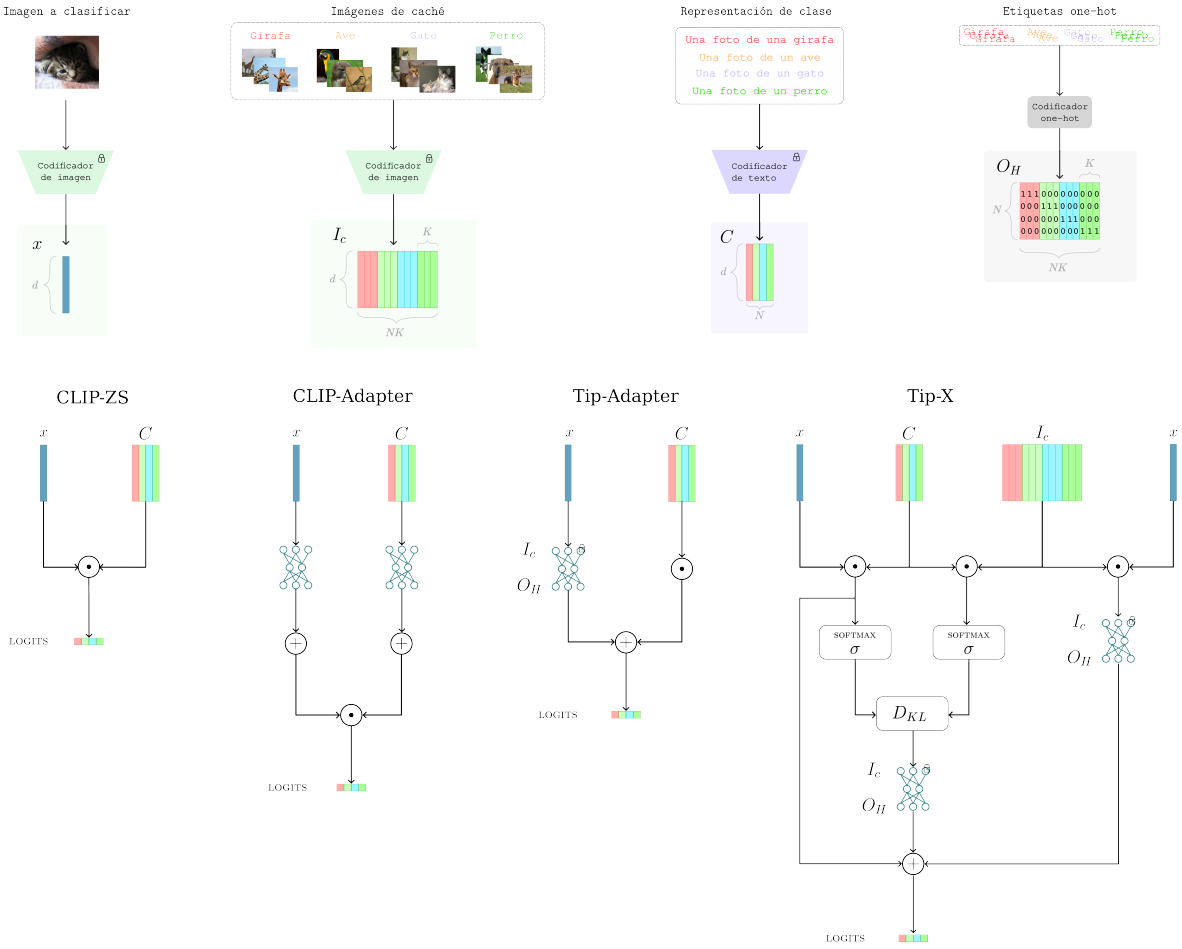

Figura B.2: Representación de los diversos métodos de adaptación.

Finalmente, este último método de adaptación se basa en la premisa de no tener calibrado el espacio vectorial entre imágenes, Vishaal Udandarao explica detalladamente en el capítulo 5.2 de [24] las causas y consecuencias de este hecho. A continuación se procede a resumir este efecto sin entrar en gran detalle. Se dice que el espacio de representación de las imágenes no se encuentra calibrado para comparativas intermodales debido a que nunca ha sido entrenado de forma explícita para realizar dicha tarea. *CLIP* únicamente optimiza explícitamente la proyección próxima entre las representaciones entre imagen y texto, además el espacio de texto presenta ciertas ventajas ya descritas como la jerarquía o que la probabilidad de tener la misma descripción de texto[3] es mucho mayor que la probabilidad de tener dos imágenes iguales. De esta forma, se tiene que el espacio vectorial asociado a la imagen presenta relaciones entre representaciones intermodales mucho más débiles que el del texto y sobre todo que

[3]Muchas de las descripciones con las que se entrena *CLIP* se generan de forma automática a partir de conjuntos de datos para clasificación en los que se cambia el texto introductorio.

el conjunto. Para respaldar esta hipótesis, se analizan los histogramas asociados a los productos entre representaciones vectoriales tanto intramodales como intermodales. La distribución de las similitudes coseno entre vectores intermodales, presenta una varianza y media bajas (una media baja es indicativo de separabilidad). Por otro lado, se tiene que la distribución de las similitudes coseno entre representaciones latentes intramodales presenta, todo lo contrario, una varianza y media altas. Adicionalmente, para respaldar las afirmaciones anteriores se observa como la media en el espacio de texto es mucho mayor que en el de la imagen. Este hecho directamente concuerda con la probabilidad de ocurrencia de un mismo elemento, mucho más probable en el caso de un texto que una imagen.

Como solución se propone el adaptador *Tip-X* [24], de forma que este hace uso de las distancias intramodales de una forma calibrada. En primer lugar, de forma análoga al resto de métodos, se obtienen los productos escalares entre: el vector de la imagen a buscar y los vectores que representan cada clase $F_x = x^T \cdot C$ y luego entre los vectores que representan cada clase con los ejemplos disponibles en la caché $F_c = C^T \cdot I_c$, dado que se desea calibrar los puntajes se obtienen las denominadas firmas que caracterizan a cada ejemplo mediante el uso de la función `softmax`, de esta forma se tienen las firmas $s_x = \sigma(F_x)$ y $s_c = \sigma(F_c)$. Dado que ahora se tratan de pseudo distribuciones de probabilidad, se mide la similitud entre estas muestras mediante el uso de la divergencia KL. Concretamente, se tiene que modificar el signo dado que una medida de similitud es inversamente proporcional al producto escalar y además, se realiza un escalado para que se encuentre en el mismo rango. Además de este término novedoso, se incorpora en el resultado final un término asociado a la similitud entre la búsqueda con el banco de imágenes, al igual que en *Tip-Adapter* [29], $\alpha g(x^T I_c) O_H$, con $g(x) = e^{-\beta(1-x)}$. Finalmente, se incorpora el término de búsqueda entre la imagen y los vectores asociados al texto cada clase, así se tiene que la búsqueda presenta la siguiente expresión: $x^T \cdot C + \alpha g(x^T I_c) O_H + \gamma \Psi(-KL(s_x || s_c)) O_H$ con γ otro parámetro a ajustar indicando la fuerza de la componente de la divergencia KL.

B.2.2. *CyCLIP* y la consistencia intermodal

Tal y como se ha mostrado anteriormente, no se optimiza directamente la relación entre los vectores de una misma modalidad, lo que conlleva la necesidad de calibrar un espacio si se desean usar los puntajes intramodales. En lugar de hacer uso de los adaptadores, Shashank, G. et al. [28], proponen añadir términos a la función de coste de forma que se calibren las distancias intramodales. Se definen dos términos asociados a aumentar la consistencia inter-intra modal:

– La **consistencia intermodal**: se reduce la diferencia entre el valor de los productos coseno asociados a las representaciones **no emparejadas** entre imagen y texto. De esta forma se optimiza que las representaciones vectoriales de textos e imágenes se reduzcan tanto correctas como incorrectas.

$$\mathcal{L}_{Inter} = \frac{1}{N}\sum_{j=1}^{N}\sum_{k=1}^{N}(\langle x_j, y_k\rangle - \langle x_k, y_j\rangle)^2 \tag{B.1}$$

– La **consistencia intramodal**: se reduce la diferencia entre el valor de los productos coseno asociados a las representaciones **no emparejadas** entre la misma modalidad, tanto imágenes como textos. Esto fuerza a compactar una modalidad de forma que se regularizan las distancias intramodales.

$$\mathcal{L}_{Intra} = \frac{1}{N}\sum_{j=1}^{N}\sum_{k=1}^{N}(\langle x_j, x_k\rangle - \langle t_k, t_j\rangle)^2 \tag{B.2}$$

Finalmente, estos términos se incorporan a la función de coste original de *CLIP* , descrita en el algoritmo 1, mediante un par de parámetros de ajuste (λ_1 y λ_2):

$$\mathcal{L}_{CyCLIP} = \mathcal{L}_{CLIP} + \lambda_1\mathcal{L}_{Inter} + \lambda_2\mathcal{L}_{Intra} \tag{B.3}$$

Además de esto, Shashank, G. et al., definen una métrica de consistencia asociada a la sincronía entre las etiquetas predichas entre la búsqueda intermodal con las predichas mediante los vectores intramodales.

$$Consistencia_k = \frac{1}{N}\sum_{j=1}^{N}\left[P_I^k(I_j) = P_T(I_j)\right] \tag{B.4}$$

Entendiendo por $P_T(I_j)$ la etiqueta asociada a la imagen j y a P_I^k como la etiqueta predicha a partir de la mayoría entre las k imágenes más cercanas. En nuestro análisis no incorporaremos este tipo de métrica debido a que el problema que se analiza en mayor profundidad es el de recuperación de información, no clasificación. Sin embargo, si se analiza el efecto de la función de coste propuesta en *CyCLIP* [28] y el posible beneficio para mejorar esta tarea.

Anexos C

Diseño de la interfaz de pruebas

Se trata de una interfaz desarrollada íntegramente con *Plotly* y *Python*. Mediante componentes HTML, se generan los vectores con las propiedades deseadas. Posteriormente, se realizan las llamadas a funciones que calculan tanto los parámetros intermodales como el valor de las funciones de pérdidas.

- A: número de muestras a simular.
- B: mantener la semilla de generación fija.
- C: ponderación de los hiperparámetros α y β.
- D: selectores del ángulo y concentricidad para las distribuciones generadoras de texto e imagen.
- E: valor de la función de pérdidas asociado a la configuración actual.
- F: visualización de los vectores de texto e imagen.
- G: valores de brecha intermodal, desalineamiento y agrupamiento asociados a la configuración actual.

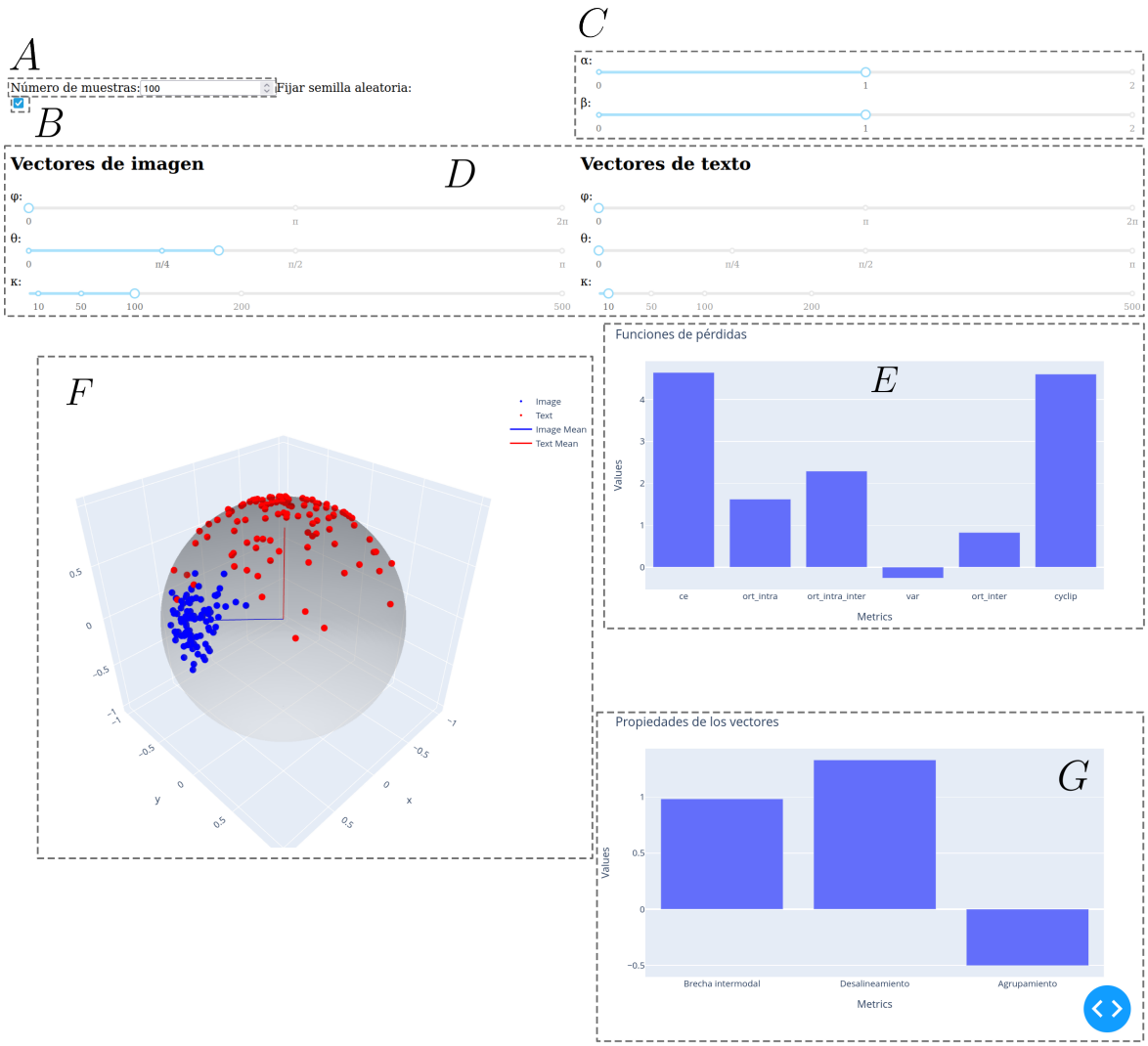

Figura C.1: Interfaz para la visualización de las funciones de pérdidas con distintas configuraciones espaciales.

Anexos D

Resultados de los entrenamientos

D.1. Pictogramas ARASAAC en inglés

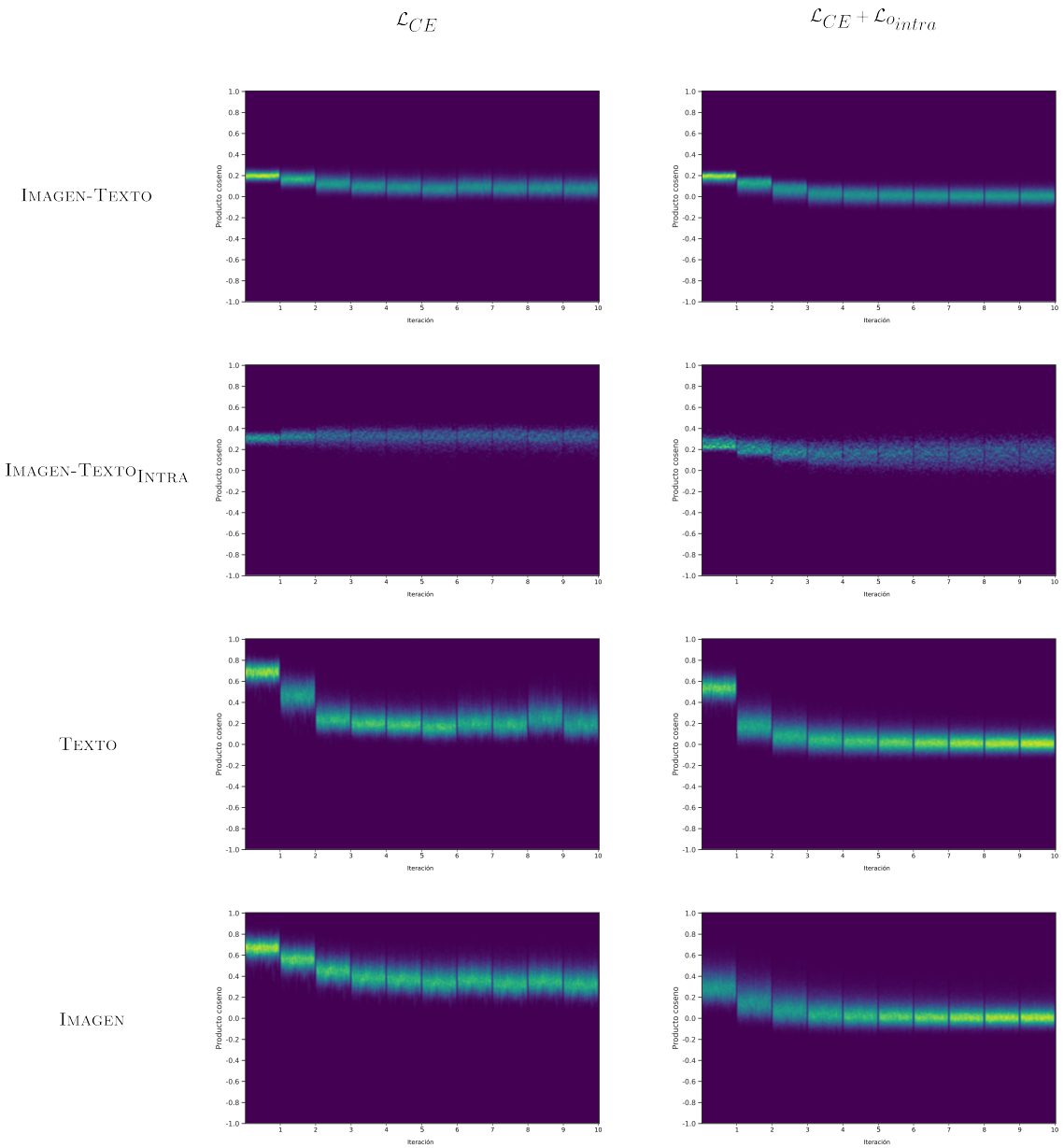

Figura D.1: Evolución de los histogramas distancias.

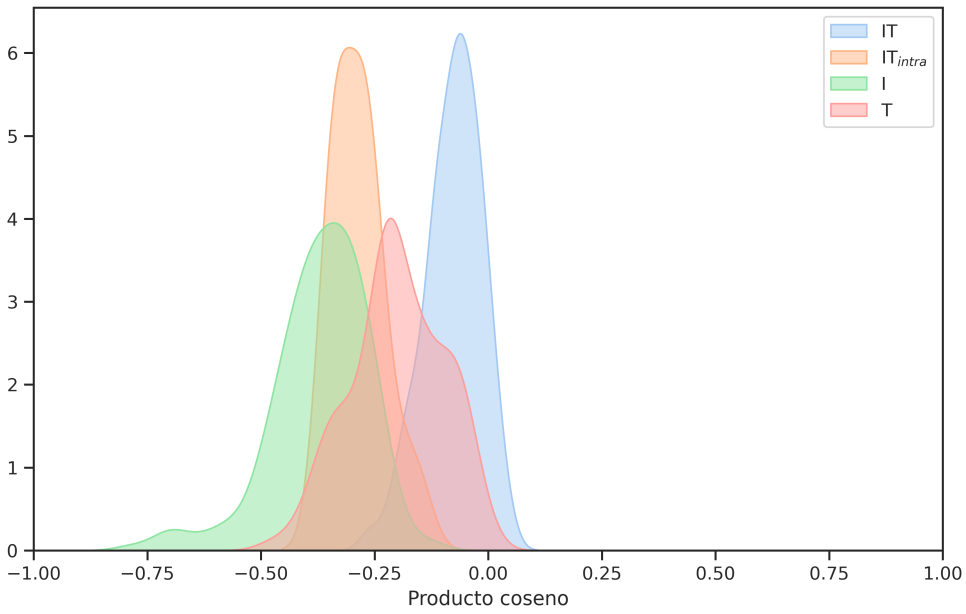

Figura D.2: Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE}$.

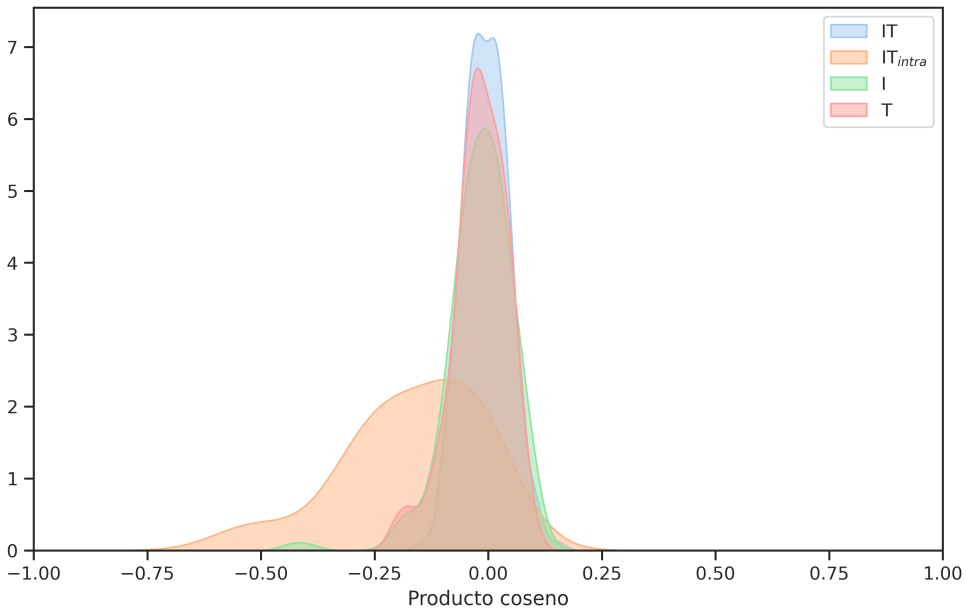

Figura D.3: Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE} + \mathcal{L}_{o_{intra}}$.

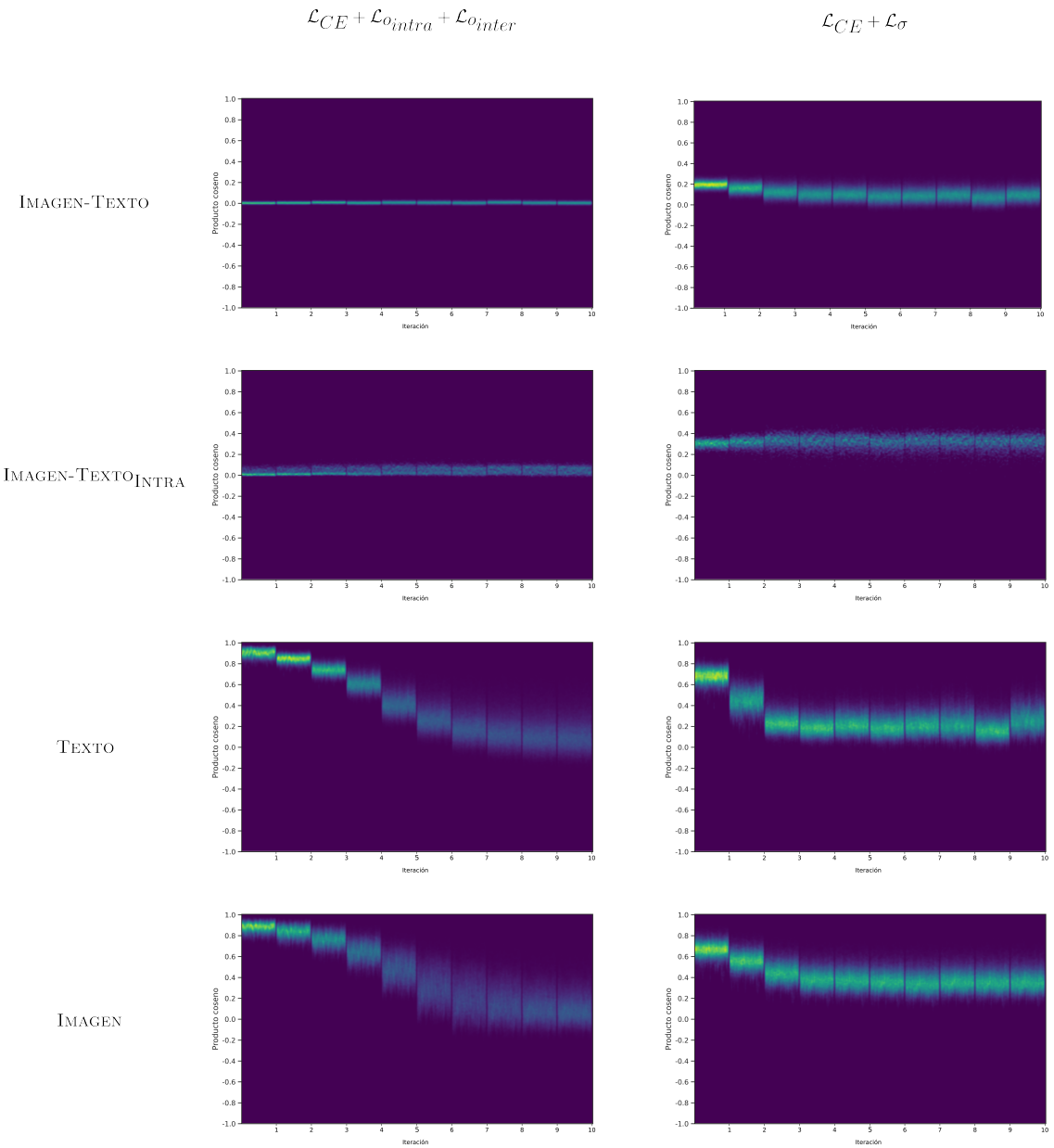

Figura D.4: Evolución de los histogramas distancias.

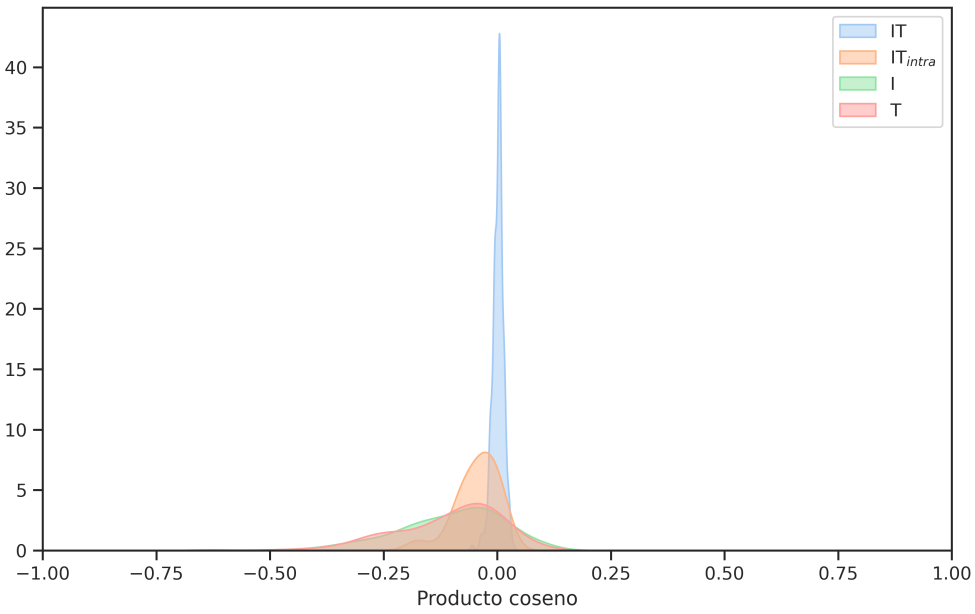

Figura D.5: Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE} + \mathcal{L}_{o_{intra}} + \mathcal{L}_{o_{inter}}$.

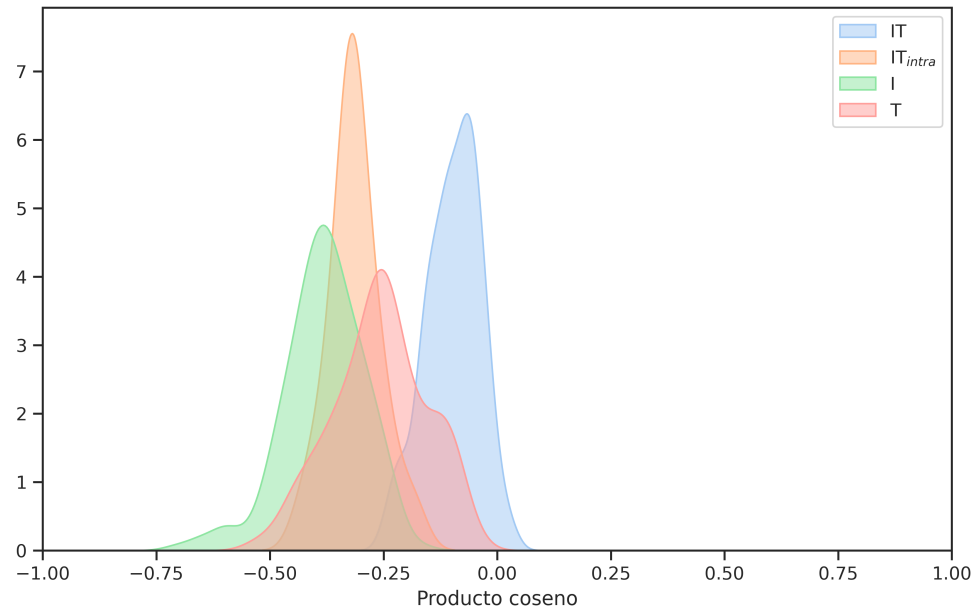

Figura D.6: Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE} + \mathcal{L}_{\sigma}$.

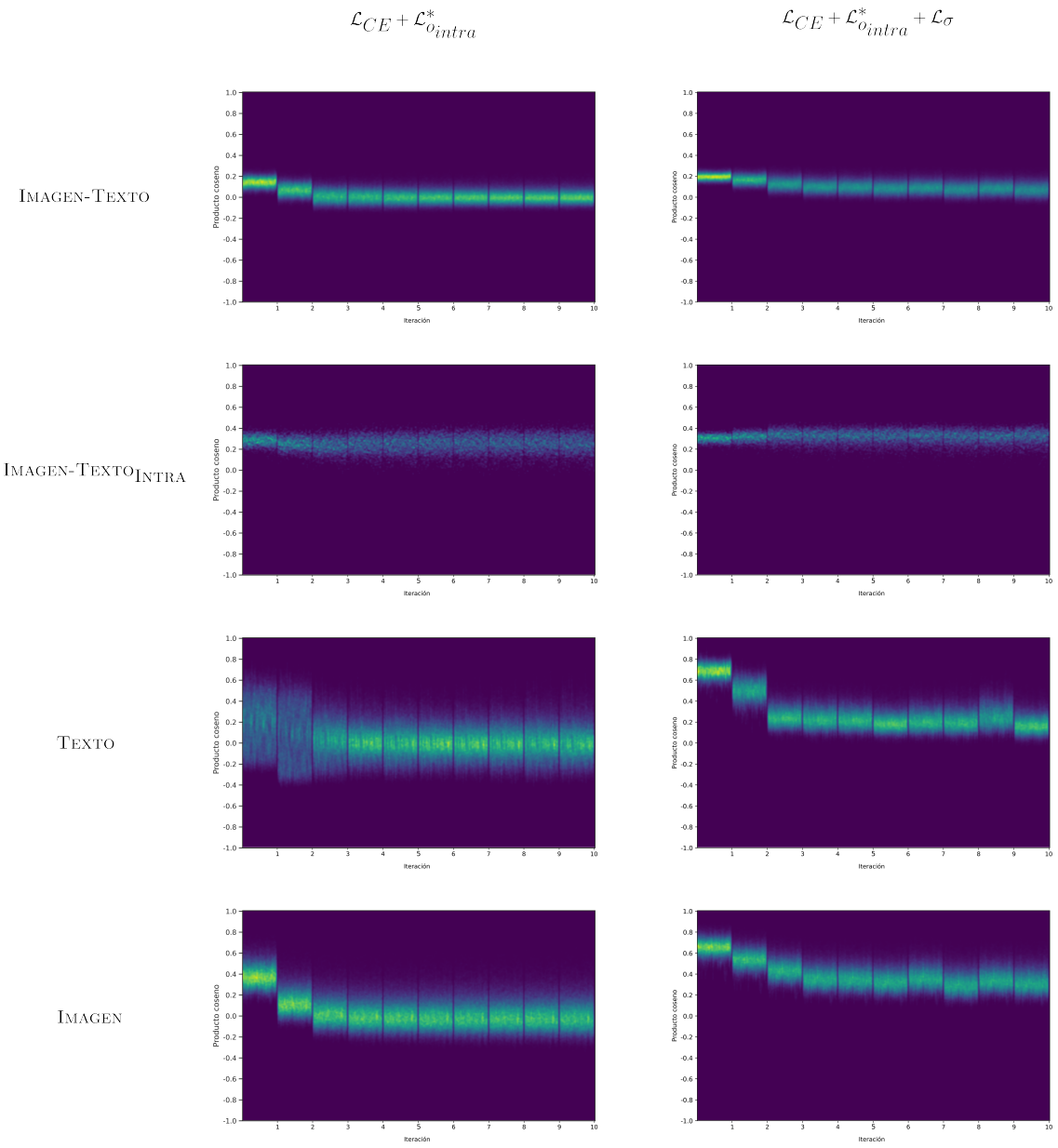

Figura D.7: Evolución de los histogramas distancias.

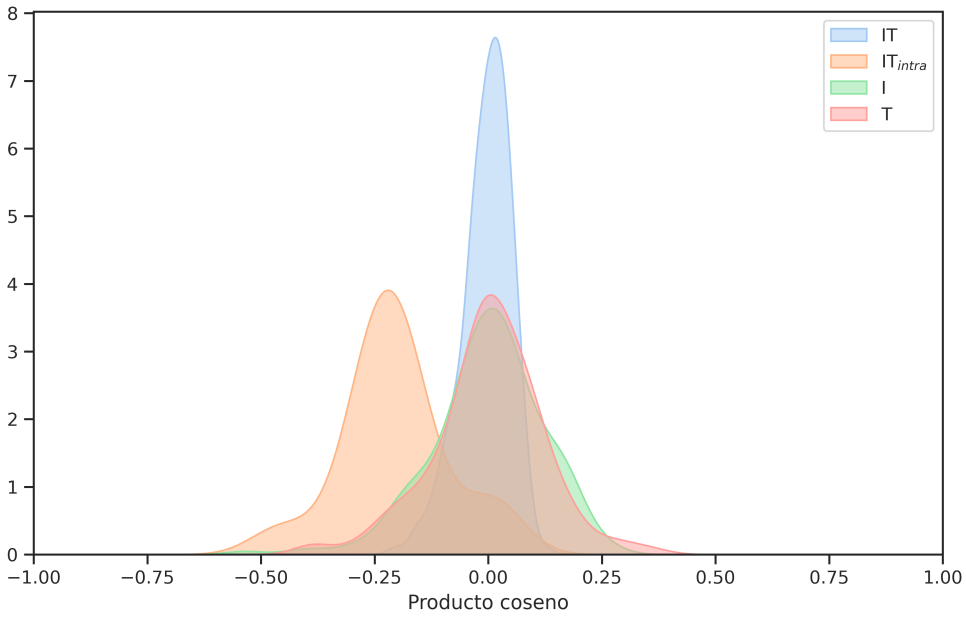

Figura D.8: Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE} + \mathcal{L}^*_{o_{intra}}$.

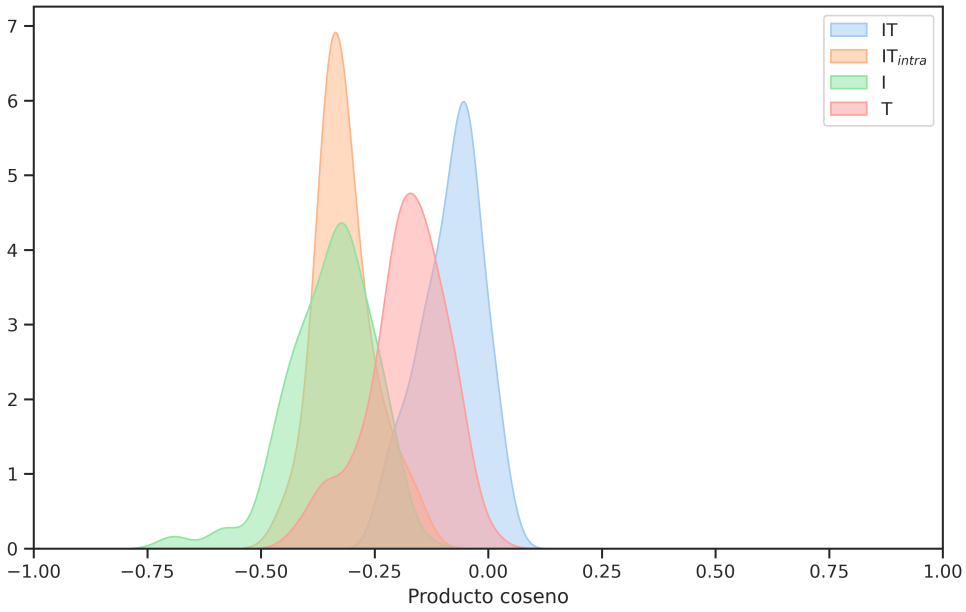

Figura D.9: Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE} + \mathcal{L}^*_{o_{intra}} + \mathcal{L}_{\sigma}$.

D.2. Pictogramas ARASAAC en español

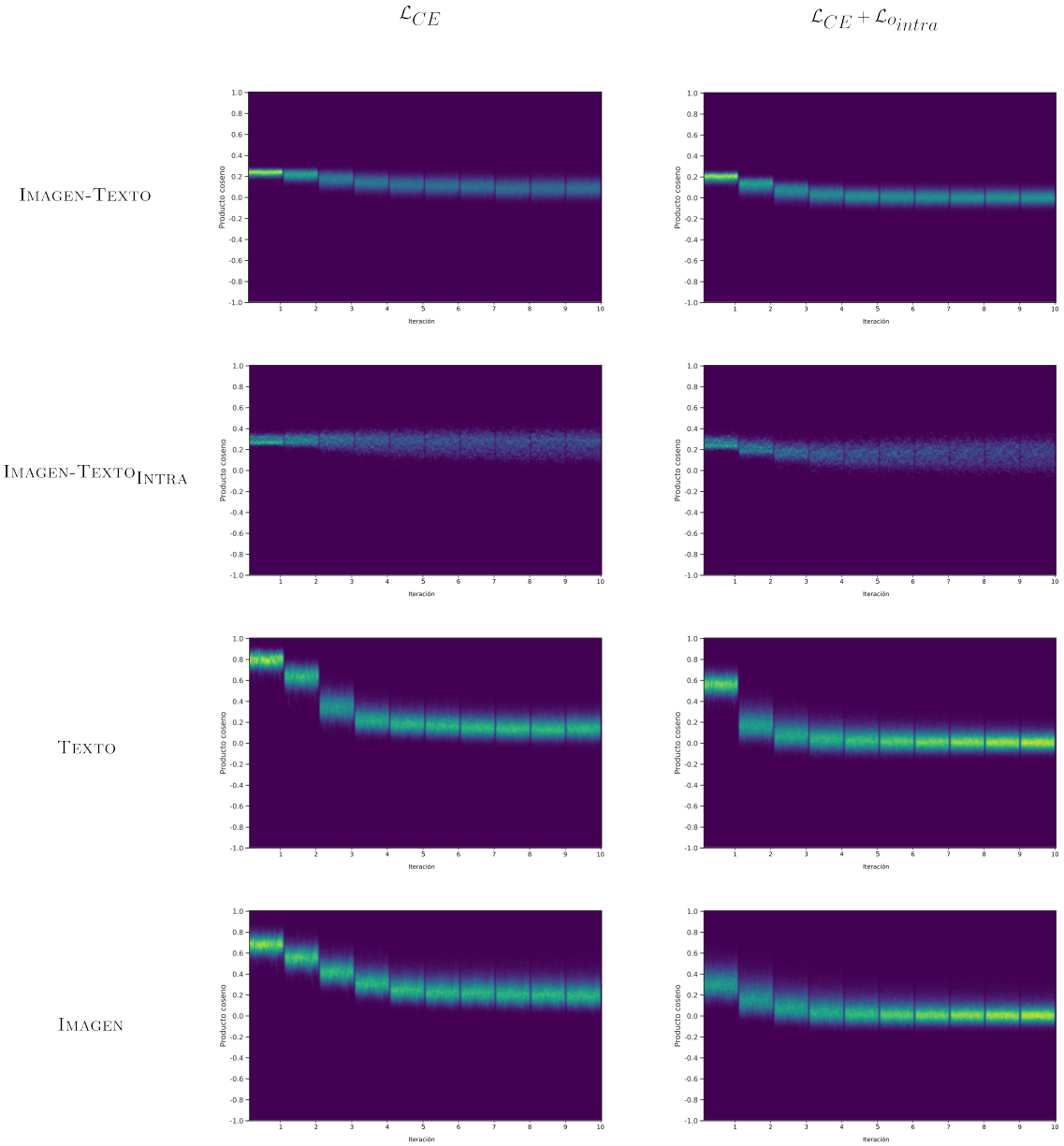

Figura D.10: Evolución de los histogramas distancias.

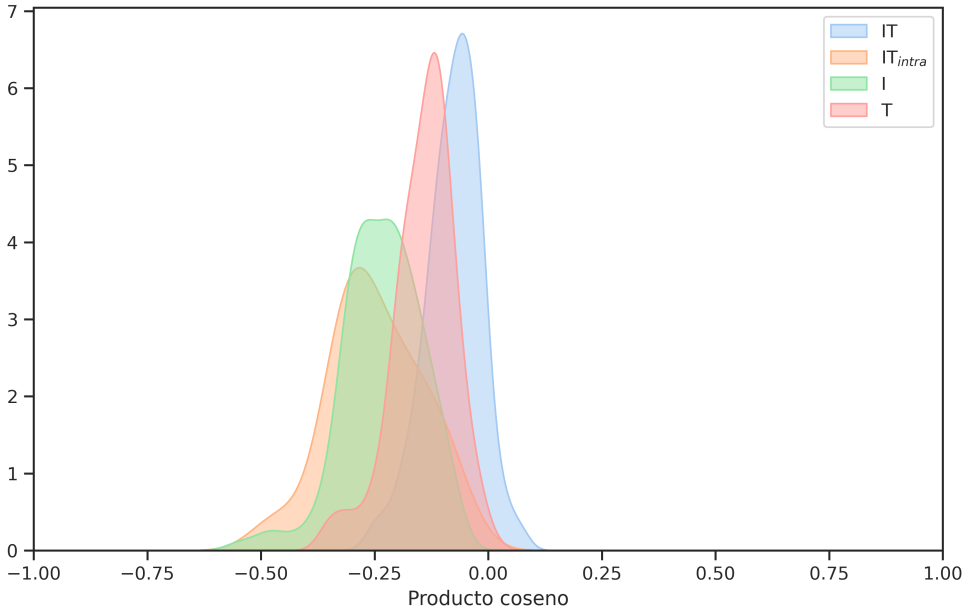

Figura D.11: Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE}$.

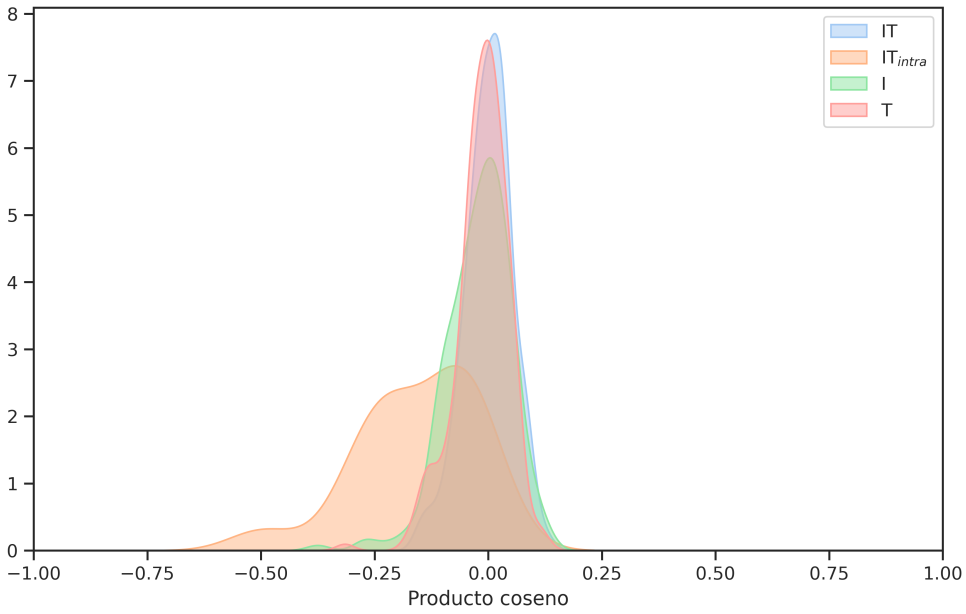

Figura D.12: Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE} + \mathcal{L}_{o_{intra}}$.

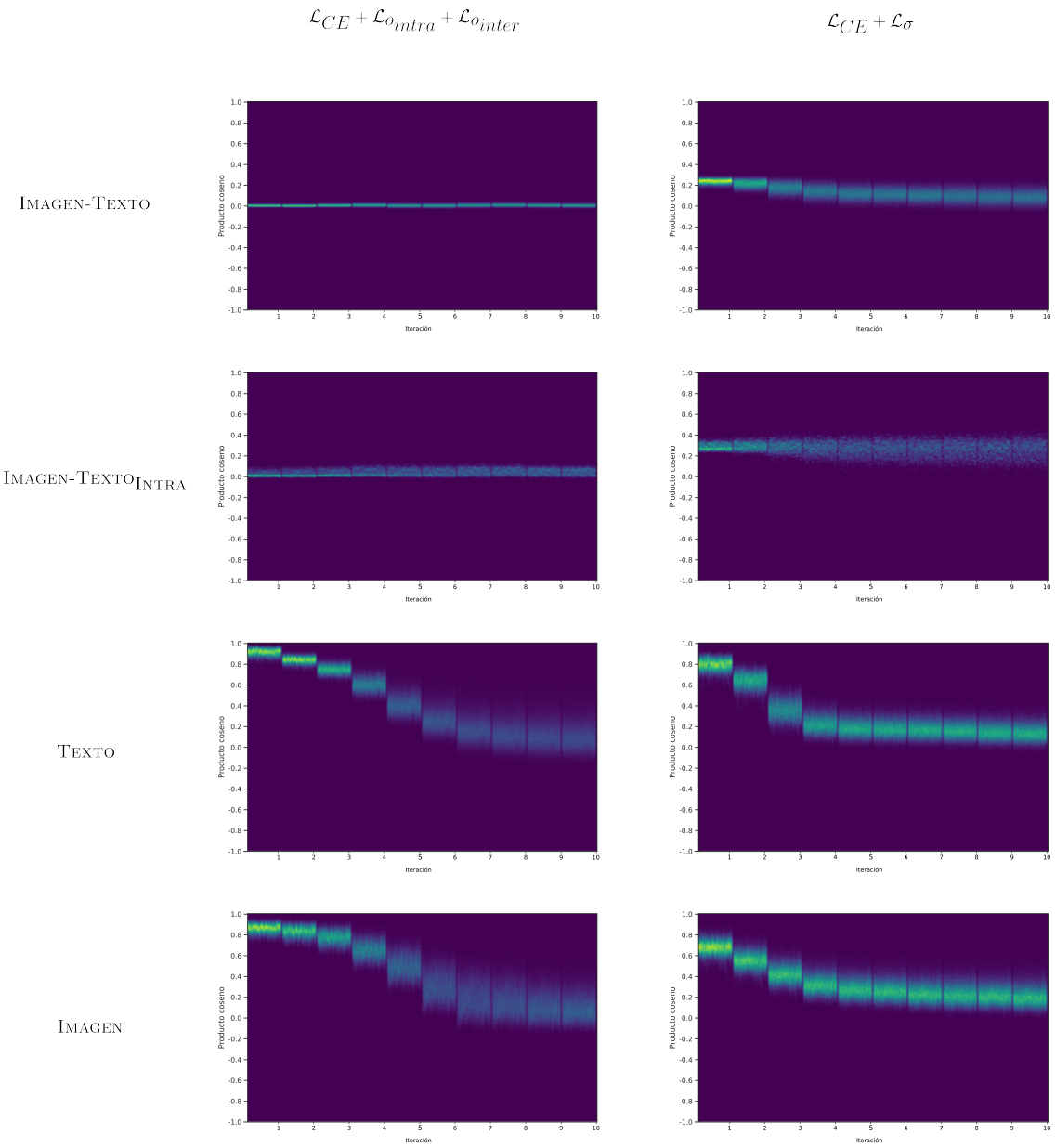

Figura D.13: Evolución de los histogramas distancias.

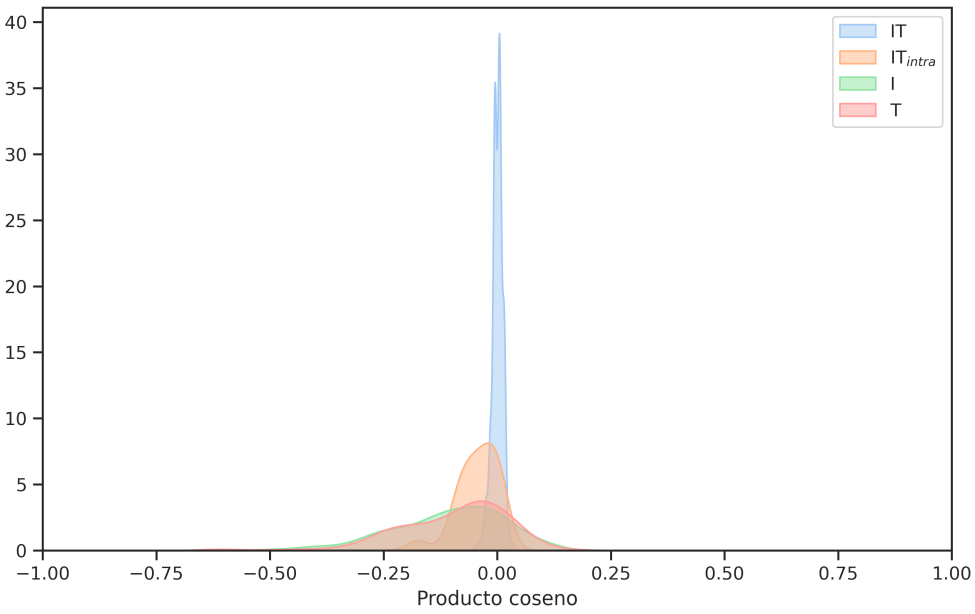

Figura D.14: Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE}+\mathcal{L}_{o_{intra}}+\mathcal{L}_{o_{inter}}$.

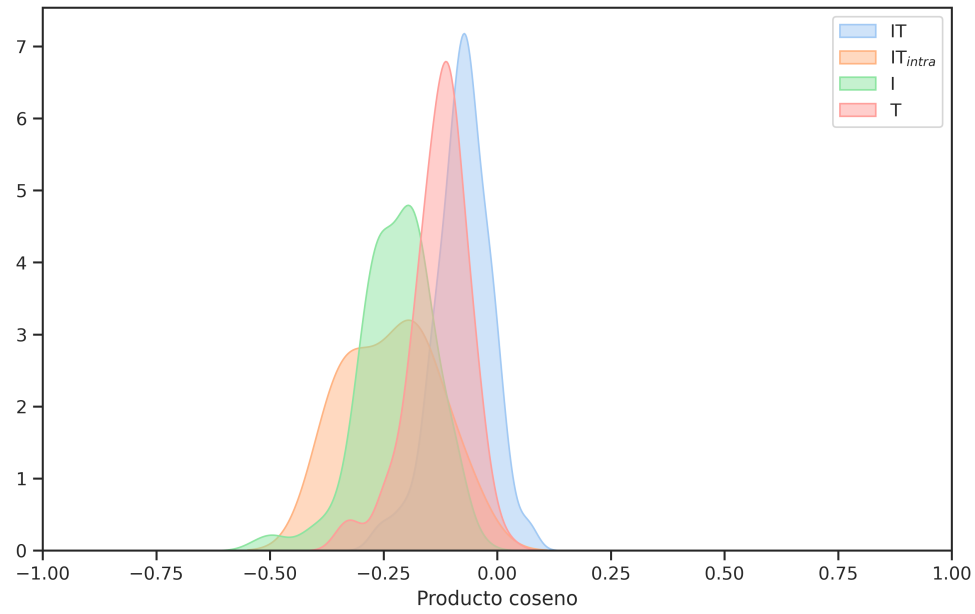

Figura D.15: Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE}+\mathcal{L}_{\sigma}$.

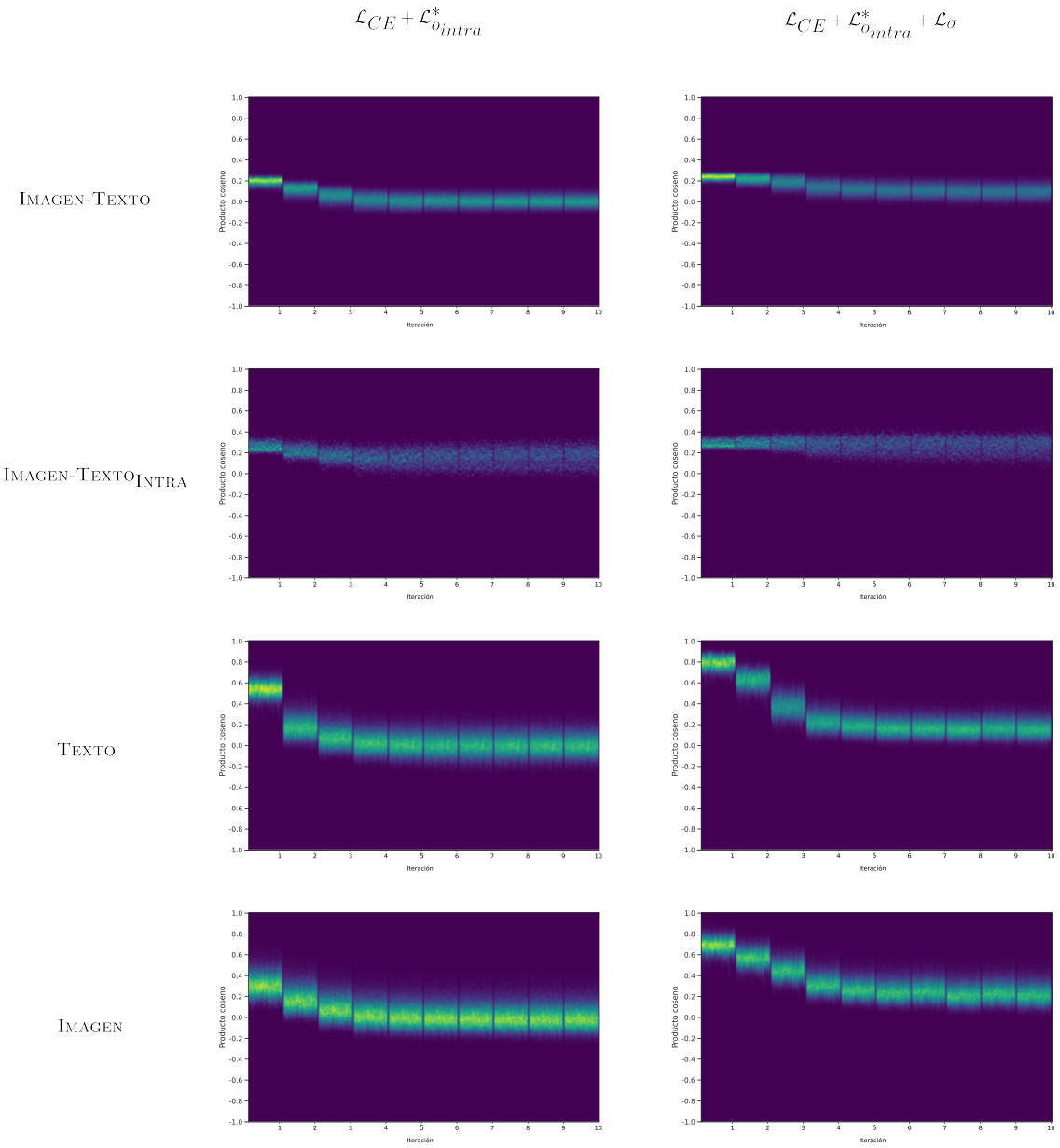

Figura D.16: Evolución de los histogramas distancias.

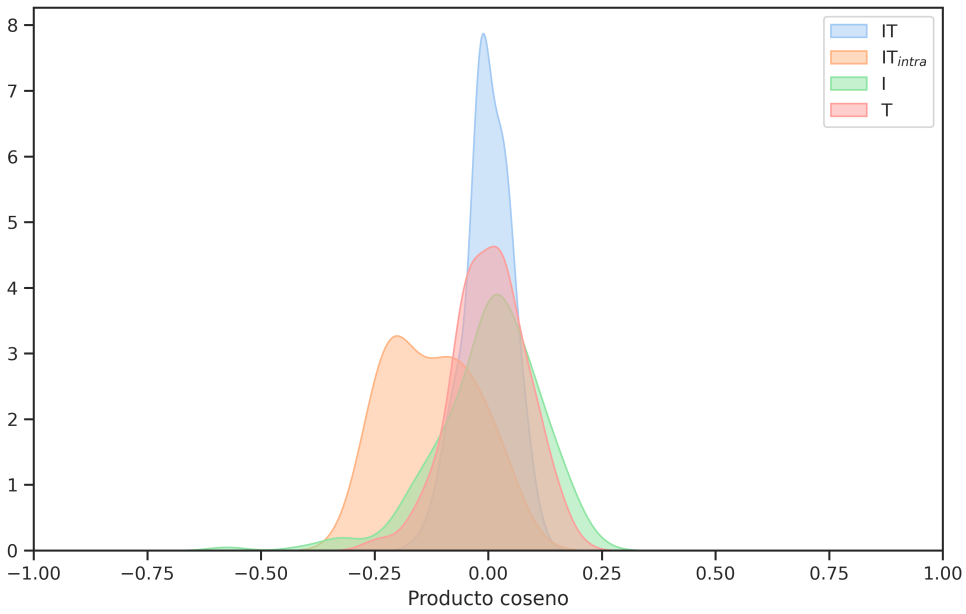

Figura D.17: Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE} + \mathcal{L}^*_{o_{intra}}$.

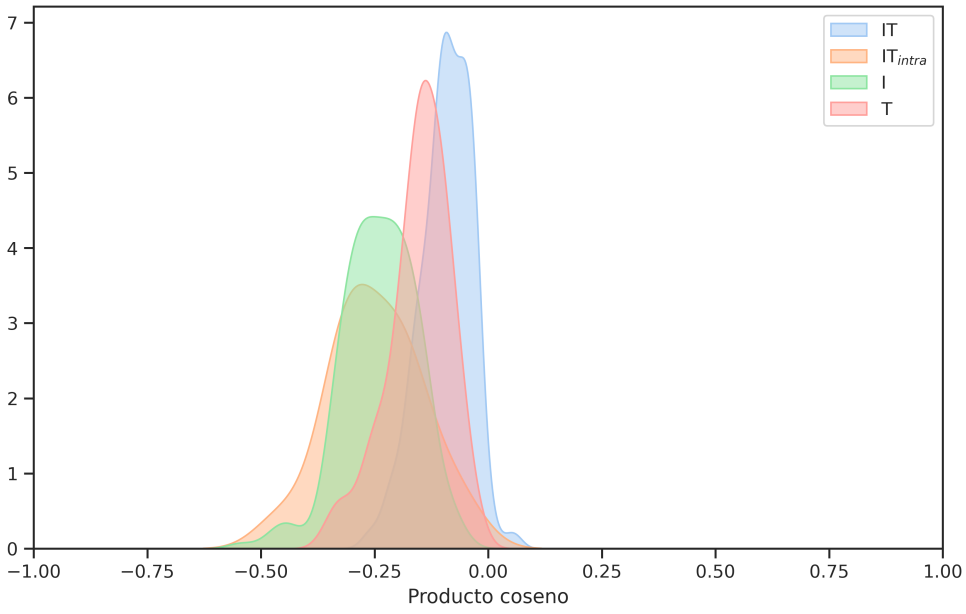

Figura D.18: Histogramas de distancias tras el entrenamiento con $\mathcal{L}_{CE} + \mathcal{L}^*_{o_{intra}} + \mathcal{L}_{\sigma}$.

Anexos E

Los espacios altamente dimensionales no son intuitivos

A continuación se presentan las ecuaciones del volumen contenido tanto para una hiperesfera como para un cubo d dimensional. Para el hipercubo de lado $2r$, y la hiperesfera[1] de radio r se tiene:

$$V_{hipercubo} = (2r)^d \tag{E.1}$$

$$V_{hiperesfera} = \frac{2r^d \pi^{d/2}}{d\Gamma(d/2)} \tag{E.2}$$

Si se expresa el cociente entre volúmenes y se hace crecer la dimensión, d se tiene:

$$\lim_{d\to\infty} \frac{V_{hipercubo}}{V_{hiperesfera}} = \frac{d2^{d-1}\Gamma(d/2)}{\pi^{d/2}} \to \infty \tag{E.3}$$

Con la ecuación E.3, se tiene que el volumen comprendido por la hiperesfera resulta insignificante respecto al del cubo, hecho contra intuitivo a lo observado en tres dimensiones.

Siguiendo en esta línea, si se desea interpretar el volumen de un espacio con un número elevado de dimensiones, es conveniente pensar en distribuciones de probabilidad. Se elige un punto aleatorio en el hipercubo con lado 2, mediante una distribución uniforme d dimensional, es decir, de -1 a 1 en cada dimensión. Encontrarse cerca de una esquina requiere que todas las variables aleatorias sean próximas a 1 o -1, hecho poco probable. Sin embargo, encontrarse cerca de una cara responde a que únicamente una de todas estas variables aleatorias, se encuentre próxima a 1 o -1. Se puede observar como este último hecho resulta mucho más probable que el anterior. De esta forma, se tiene que la mayor parte del volumen de un hipercubo se encuentra en su frontera.

[1] Siendo Γ la función gamma de Euler.