UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

MODELO AUTOSUPERVISADO PARA RECOMENDACIÓN VISUAL PARA ECOMMERCE DE MODA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

ROBERTO ARAYA DAY

PROFESOR GUÍA:
JOSÉ SAAVEDRA RONDO

MIEMBROS DE LA COMISIÓN:
BENJAMÍN BUSTOS CÁRDENAS
RODRIGO VERSCHAE TANNENBAUM

SANTIAGO DE CHILE
2023

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL
EN COMPUTACIÓN
POR: ROBERTO ARAYA DAY
FECHA: 2023
PROF. GUÍA: JOSÉ SAAVEDRA RONDO

MODELO AUTOSUPERVISADO PARA RECOMENDACIÓN VISUAL PARA ECOMMERCE DE MODA

El problema de *fashion compatibility* radica en determinar la compatibilidad entre un conjunto de prendas. Este desafío posee gran relevancia y puede ser aplicado en diversos contextos, como plataformas de comercio electrónico de moda y aplicaciones de vestuario. En tales situaciones, una interfaz capaz de sugerir productos compatibles a los usuarios incrementaría los ingresos, mejoraría la experiencia de compra, y aumentaría la retención de clientes.

Se han propuesto soluciones de inteligencia artificial para abordar la *fashion compatibility* mediante modelos supervisados. Estos modelos se entrenan con tripletes de datos: una prenda de referencia, una compatible y una incompatible. A pesar de los resultados positivos, presentan debilidad en los ejemplos negativos de incompatibilidad, seleccionados al azar, lo que puede llevar a aprendizaje no deseado. Esto podría generar recomendaciones erróneas al considerar características irrelevantes, resultando en sugerencias carentes de pertinencia o, peor aún, contraproducentes para la experiencia del usuario y la confianza en la plataforma.

Así, es altamente deseable desarrollar modelos de aprendizaje que utilicen exclusivamente pares de prendas compatibles. Esto eliminaría la necesidad de abordar la complejidad asociada con la selección de ejemplos negativos. En el contexto de esta tarea, se exploran dos modelos que han demostrado generar representaciones de alta calidad en tareas como la clasificación de imágenes y la detección de objetos usando solo ejemplos positivos.

Este trabajo realiza un estudio exhaustivo de los modelos de recomendación de prendas fundamentados en imágenes. A continuación, se proponen modelos de recomendación en el contexto de su implementación en una plataforma de comercio electrónico de moda. Estos modelos se someten a evaluación utilizando el conjunto de datos *Polyvore Outfits*, un conjunto de datos diseñado para evaluar tanto la compatibilidad como la recuperación de prendas. Por último, se crea una interfaz web que permite visualizar los resultados obtenidos por los modelos.

Después de rigurosas pruebas y comparaciones, se ha constatado que los ejemplos negativos, aunque puedan no ser completamente exactos, proporcionan información relevante y posibilitan el aprendizaje de sistemas capaces de generar recomendaciones más precisas en comparación con aquellos basados únicamente en ejemplos positivos. Estos ejemplos permiten al sistema comprender y diferenciar con mayor precisión las disparidades entre prendas compatibles e incompatibles, lo cual contribuye a una coherencia y robustez superiores en las recomendaciones generadas.

Agradecimientos

Quiero agradecer a mi familia por su apoyo incondicional a lo largo de mi recorrido universitario y educativo. Quiero agradecer especialmente a mi padre, quien me ha brindado innumerables oportunidades para seguir aprendiendo y ha sido una fuente constante de motivación para que profundice mis conocimientos en el área. También quiero agradecer a mi madre, quien me ha apoyado y ha sido un pilar fundamental para desarrollarme como persona e ingeniero con calma y serenidad. Asimismo, también quiero mencionar a mis hermanos, con quienes comparto todos los días y cuyo apoyo me permite seguir adelante y mejorar como persona.

Por último, también debo agradecer a mi profesor guía José Saavedra, quien merece un reconocimiento por su notable paciencia y dedicación que ha destinado para este proyecto.

Tabla de contenido

Índice de ilustraciones

Índice de tablas

Capítulo 1

Introducción

Con el crecimiento de las compras en línea, los sistemas de recomendación han cobrado cada vez más relevancia para guiar el interés del cliente y obtener ingresos en aplicaciones de comercio electrónico *ecommerce*. Un problema particular en los servicios de venta de ropa es determinar si un conjunto de prendas son compatibles y pueden combinarse para formar un estilo coherente. Este desafío, conocido como compatibilidad de moda (o *fashion compatibility* en inglés), ha sido objeto de estudio importante en los últimos años utilizando técnicas de inteligencia artificial. Resolver este problema implica capturar un sentido subjetivo humano para relacionar prendas de moda y considerar factores como color, textura, estilo y funcionalidad en conjunto. La exploración de soluciones efectivas en este campo es esencial para mejorar la experiencia del usuario y aumentar la satisfacción en el ámbito de la moda en línea.

Un sistema así podría ser utilizado en una plataforma de venta de ropa para sugerir prendas compatibles con las que se están comprando o en las que el cliente ha mostrado interés. El sistema de recomendación podría generar ingresos a las plataformas puesto que puede despertar el interés de los clientes y comprar las sugerencias. Esto resulta en un beneficio para los minoristas y empresas de moda, como también a los consumidores al sentirse mas seguros y satisfechos con las recomendaciones de moda. Es importante destacar la dificultad del problema de *fashion compatibility* debido a la compleja interacción de la creatividad humana, la experiencia en estilo y la auto-expresión involucradas en el proceso de transformar una colección de artículos aparentemente inconexos en un concepto cohesivo.

En el campo de la inteligencia artificial (IA), se han propuesto numerosas soluciones que emplean modelos de aprendizaje supervisado, así como algunos de auto-supervisión. Los modelos de aprendizaje tienen la capacidad de ajustar sus parámetros mediante el uso de datos para abordar tareas específicas, mejorando sus resultados predictivos en cada iteración. En particular, los modelos supervisados emplean valores de uno o varios campos de entrada para predecir uno o varios resultados. Esto implica que para entrenarlos se requieren datos con un valor a predecir, al cual llamaremos y. Por lo general, estos valores etiquetados deben ser asignados manualmente, involucrando trabajo humano.

En contraste, los modelos auto-supervisados representan un enfoque donde el propio modelo crea sus objetivos de entrenamiento a partir de los datos disponibles, evitando la necesidad de etiquetas específicas y para cada instancia. En lugar de basarse en salidas conocidas, los modelos auto-supervisados aprovechan la estructura inherente de los datos para establecer sus propios objetivos de entrenamiento. Esta estrategia puede potenciar la eficiencia del proceso de aprendizaje y ampliar las posibilidades de aplicaciones en situaciones donde los datos etiquetados son limitados o costosos de adquirir.

Los modelos supervisados que han alcanzado el rendimiento más destacado en la tarea de *fashion compatibility* incluyen Bi-LSTM [21], *Siamese Networks* [1], SCE-Net [15] y CSA-Net [22]. Estos modelos utilizan redes convolucionales (CNN) para extraer características de las imágenes y mapearlas en un espacio de representación. Luego, cada modelo establece condiciones de similitud, o "*similarity conditions*", para definir la compatibilidad entre las prendas. En algunos casos, estas condiciones se definen manualmente, a menudo basadas en textura, color o forma de las prendas, mientras que en otros casos se aprenden automáticamente. Los modelos generan vectores condicionales que luego se transforman para obtener un espacio final de vectores, donde se consideran prendas compatibles si están cerca en dicho espacio.

Por el otro lado, los modelos auto-supervisados que han demostrado el mejor rendimiento en la tarea de *fashion compatibility* incluyen STOC [4] y el enfoque presentado por Aditya Popli y colaboradores [2]. STOC, el primer modelo, se centra en aprender a vectorizar imágenes con colores y patrones de textura similares en proximidad. Antes de su desarrollo, los modelos auto-supervisados se centraban principalmente en la forma de las prendas, lo que resultaba en un rendimiento limitado. En contraste, el segundo modelo auto-supervisado aprende representaciones de parches en lugar de la imagen completa, permitiéndole aprender que las representaciones de diferentes prendas vestidas por la misma persona están más cercanas en comparación con aquellas de personas diferentes.

En particular, la mayoría de los modelos que logran los mejores resultados se entrenan utilizando tripletas de datos. Esto implica una prenda de referencia, una prenda compatible y una prenda incompatible, y utilizan las categorías de las prendas para su entrenamiento. La prenda de referencia se asocia con una categoría c_i, mientras que las prendas compatibles e incompatibles se asocian con una categoría c_j. Por ejemplo, la imagen de referencia podría ser una camisa, mientras que las prendas compatibles e incompatibles podrían ser pantalones.

Las imágenes de referencia y las prendas compatibles se obtienen de catálogos de ropa donde se encuentran en conjunto, mientras que las prendas incompatibles se seleccionan de forma aleatoria. En otras palabras, se elige una prenda al azar del conjunto de datos para que actúe como la prenda incompatible. No obstante, esta estrategia presenta un problema significativo: la aleatoriedad puede resultar en ejemplos que en realidad son compatibles, lo que introduce un sesgo sustancial en las características y puede llevar a recomendaciones erróneas. Si una prenda negativa seleccionada al azar resulta ser, en realidad, compatible con la prenda de referencia, el modelo puede aprender características incorrectas sobre lo que constituye la incompatibilidad. Esto distorsiona el proceso de aprendizaje al introducir información errónea en el modelo.

Por lo tanto, es de suma importancia desarrollar estrategias más robustas y efectivas para la selección de ejemplos negativos en la formación de estos modelos. Además, resulta prometedor explorar arquitecturas que prescindan por completo del uso de ejemplos negativos en su proceso de entrenamiento. Al eliminar esta dependencia de ejemplos incompatibles, se puede reducir significativamente la posibilidad de que los modelos adquieran conceptos erróneos y, en consecuencia, se mejore la calidad de las recomendaciones generadas por los sistemas de compatibilidad de moda. En este trabajo, se evalúa el rendimiento de modelos que no requieren ejemplos negativos para su entrenamiento.

En resumen, este trabajo realiza un análisis exhaustivo de los modelos de recomendación existentes. Evalúa los requisitos necesarios para desarrollar un modelo de recomendación de ropa de alto rendimiento, especialmente diseñado para su implementación en un entorno de *ecommerce* de moda. Luego, se lleva a cabo una evaluación comparativa del rendimiento de los modelos más relevantes implementados en relación con el estado del arte, utilizando métricas específicas para esta tarea. Además, se implementa un sistema de recomendación de prendas que facilita la comparación de los resultados obtenidos por los modelos propuestos. Para cerrar este trabajo, se desarrolla una interfaz web que permite visualizar los resultados de manera efectiva.

1.1. Objetivo General

Realizar un estudio de los modelos de compatibilidad de moda y desarrollar un modelo de recomendación de ropa para el *e-commerce*. El modelo debe funcionar con una precisión similar al modelo de recomendación supervisado CSA-Net [22] utilizado actualmente, donde la recomendación debe entenderse como un proceso de recuperación de productos.

1.2. Objetivos Específicos

1. Realizar un estudio de los modelos existentes de recomendación de ropa basado en compatibilidad.
2. Definir un conjunto de datos preliminar y métricas de evaluación para medir la efectividad de los modelos de recomendación de ropa.
3. Definir y evaluar los modelos de compatibilidad de prendas en función de su aplicabilidad en el contexto de un comercio electrónico de moda, y comparar sus resultados mediante métricas de evaluación.
4. Evaluar la aplicación de modelos que utilicen solo ejemplos de pares positivos para su entrenamiento.
5. Implementar un sistema de recuperación de ropa y evaluar sus resultados. La recuperación consiste en retornar un conjunto de prendas compatibles con una imagen de entrada.
6. Construir una interfaz que permita visualizar los resultados de los modelos de recuperación de ropa.

Capítulo 2

Marco Teórico

En esta sección se proporciona contexto acerca de las tecnologías y conceptos claves que se utilizan en las próximas secciones para abordar el problema de *fashion compatibility*.

2.1. Definiciones básicas

2.1.1. Imagen

Una imagen en computación se entiende como una representación digital compuesta por una matriz de píxeles, donde cada píxel representa un punto de la imagen y contiene información sobre el color en ese punto. En una imagen de escala de grises, cada píxel contiene un valor entre 0 y 255 representando la intensidad gris en ese punto. Para este trabajo se consideran imágenes a color, en el que cada píxel se representa mediante tres valores de intensidad: uno para el color rojo, uno para el color verde y uno para el color azul. Es decir, una imagen se define como una matriz de tres dimensiones $(\text{ancho}, \text{alto}, 3)$.

En el contexto de un sistema de recomendación de ropa, se definirá una imagen como una representación en la que se pueda distinguir una prenda. Se presenta una comparación en la Figura 1.

(a) Se distingue una polera (se considera imagen).

(b) No se considera imagen.

Figura 1: Definición de imagen

2.1.2. Categoría

La categoría en el contexto de un sistema de recomendación de ropa se refiere a la clasificación o etiqueta que se asigna a una prenda para agruparla en función de sus características.

Las categorías son utilizadas por los modelos de recomendación para organizar y clasificar las prendas y ofrecer recomendaciones más precisas. En este trabajo, una imagen se acompaña de respectiva categoría, seleccionada de un listado de categorías predefinidas. El listado corresponde a:

1. *bags*
2. *shoes*
3. *jewellery*
4. *tops*
5. *bottoms*
6. *all-body*
7. *hats*
8. *sunglasses*
9. *accessories*
10. *scarves*
11. *outerwear*

2.1.3. Ejemplos positivos y negativos

Se definen los ejemplos negativos como pares de imágenes de prendas de distintas categorías que no combinan según una clasificación previa. Por otro lado, se denominan ejemplos positivos a aquellas imágenes de prendas que sí combinan según un criterio determinado.

Se presentan algunos ejemplos en la Figura 2. En (a) se observan prendas compatibles de las categorías de bolsos y zapatos, mientras que en (b) se muestra una mochila y zapatos que combinan. En (c) y (d) se observan combinaciones de pantalones y abrigos, y gorras y bolsas que no combinan.

2.2. Inteligencia Artificial

La inteligencia artificial, también conocida como IA, es una disciplina dentro del campo de las ciencias de la computación que tiene como objetivo desarrollar máquinas capaces de imitar la inteligencia humana y realizar tareas de forma autónoma [14]. La inteligencia artificial abarca diversos subcampos. En este trabajo, nos centraremos principalmente en el aprendizaje de máquinas y el aprendizaje profundo debido a su relevancia para abordar el problema de recomendación de ropa.

(a) Ejemplo positivo (b) Ejemplo positivo

(c) Ejemplo negativo (d) Ejemplo negativo

Figura 2: Ejemplos de imágenes compatibles y no compatibles

2.2.1. Aprendizaje de Máquinas

Aprendizaje de máquinas, o aprendizaje automático, tiene como objetivo desarrollar técnicas que permitan que las computadoras ”aprendan”. Donde se determinará que un algo aprende cuando su desempeño mejora con la experiencia y mediante el uso de datos [16]. El resultado del área es así un modelo para resolver una tarea dada.

Un modelo de aprendizaje automático se define como una representación matemática de un sistema o proceso que aprende de los datos y realiza predicciones sin ser programado explícitamente. Estos modelos aprenden patrones y reglas subyacentes a los datos, lo que les permite realizar predicciones o tomar decisiones en datos no vistos. En el trabajo se evalúan diversos modelos de aprendizaje para conseguir buenos resultados por lo que se revisan algunos conceptos claves para entender su funcionamiento.

Un modelo de aprendizaje de máquinas se compone por la arquitectura, la función de pérdida o método de evaluación y el optimizador.

- Arquitectura del modelo: Se refiere a la estructura y configuración del modelo, es decir, cómo se organizan y combinan las unidades de procesamiento dentro del modelo para llevar a cabo la tarea específica para la cuál fue diseñado. La arquitectura de un modelo de aprendizaje automático va a depender del problema que se está abordando. Algunos componentes comunes que forman parte de la arquitectura son los siguientes:
  - Parámetros: Los modelos de aprendizaje automático tienen parámetros que se ajustan durante el entrenamiento para que el modelo se adapte a los datos de entrenamiento.
  - Funciones de activación: Son funciones matemáticas aplicadas a las salidas de los resultados de la aplicación de los parámetros para introducir no linealidad en el modelo. Algunas funciones de activación comunes incluyen la función sigmoide, la

función ReLU (*Rectified Linear Unit*) y la función *softmax*.

La arquitectura del modelo es fundamental para el diseño y funcionamiento de los sistemas de aprendizaje automáticos utilizados en la resolución del problema de compatibilidad de moda.

- Funciones de Pérdida: Una función de pérdida, también conocida como función de costo o función objetivo, es una medida que se utiliza en aprendizaje automático para evaluar cuánto difieren las predicciones de un modelo de los valores reales de los datos de entrenamiento. Esta medida cuantifica la calidad del modelo durante el entrenamiento para ajustar sus parámetros y minimizar la pérdida. Al minimizar la pérdida, el modelo se vuelve más preciso y es capaz de realizar mejores predicciones en datos no vistos. Para entrenar un modelo de recomendación de ropa adecuado y obtener buenos resultados es necesario definir una función de pérdida apropiada. Los siguientes conceptos son relevantes:
  - Conjunto de entrenamiento o *Training set*: Es el conjunto de datos utilizado para entrenar el modelo. El modelo ajusta sus parámetros basado en este conjunto de datos.
  - Conjunto de prueba o *Test set*: Es el conjunto de datos utilizado para evaluar el rendimiento del modelo una vez que ha sido entrenado. Este conjunto no se utiliza en el proceso de entrenamiento.
  - Conjunto de Validación o *Validation Set*: Este conjunto de datos es similar al conjunto de pruebas en el sentido que tiene como objetivo evaluar el rendimiento del modelo. Su diferencia radica en que se utiliza durante el proceso de entrenamiento del modelo para detectar problemas de sobre-ajuste y ajustar los hiper-parámetros para optimizar la arquitectura del modelo.

  Las funciones de pérdida más comunes corresponden a *Cross-Entropy Loss* y *Mean Square Error*, cuyo objetivo es aprender a predecir directamente una etiqueta, un valor o una serie de valores dada una entrada. En el contexto de modelos de recomendación de ropa se emplean funciones de pérdida basadas en *Ranking Loss*, cuyo objetivo es predecir distancias relativas entre las entradas. Se describirá esta función de pérdida de manera detallada más adelante.
- Optimizador: En el área de aprendizaje automático, los optimizadores se definen como algoritmos que se utilizan para ajustar los parámetros de un modelo durante el proceso de entrenamiento. Estos algoritmos buscan minimizar una función de pérdida o maximizar una función de rendimiento, con el objetivo de encontrar los valores óptimos de los parámetros del modelo que permitan una mejor capacidad de generalización en datos no vistos.

  Los optimizadores funcionan de manera iterativa, actualizando los parámetros del modelo en cada paso de entrenamiento utilizando información del gradiente de la función de pérdida o rendimiento. El gradiente proporciona la dirección y magnitud del cambio necesario para mejorar el rendimiento del modelo.

En el trabajo se presentan algunos modelos supervisados y auto-supervisados para abordar el problema de compatibilidad de moda. Así, es importante definirlos y explicar su distinción.

Los modelos de aprendizaje supervisados basan su aprendizaje en un conjunto de datos de entrenamiento previamente etiquetados. Esto le permite al algoritmo aprender una función capaz de predecir el atributo objetivo para un conjunto de datos nuevos. Por otro lado, los modelos auto-supervisados son algoritmos que basan su proceso de entrenamiento en un juego de datos sin etiquetas. A priori no se conoce ningún valor objetivo, está dedicado principalmente a tareas de agrupamiento o segmentación, donde su objetivo es encontrar grupos similares en el conjunto de datos.

Es relevante destacar el subárea del Aprendizaje Profundo o *Deep Learning*, una rama del campo de la inteligencia artificial que se enfoca en el entrenamiento y uso de modelos de redes neuronales para llevar a cabo tareas complejas de manera automática. La premisa fundamental es que las redes neuronales artificiales con múltiples capas pueden aprender representaciones más complejas a medida que se profundiza en la estructura de la red [12].

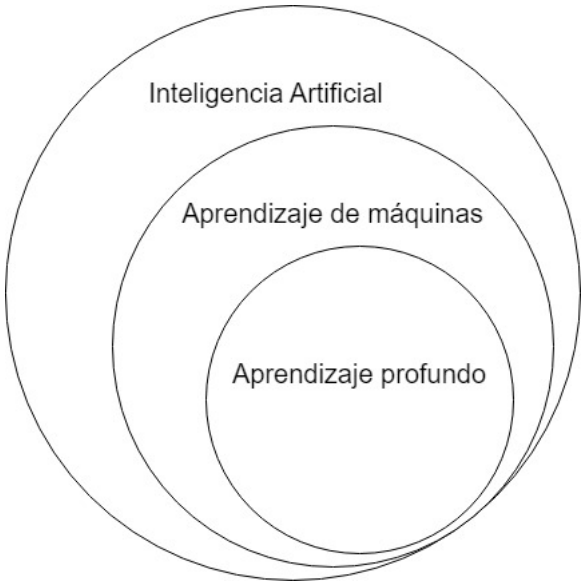

Figura 3: Áreas de inteligencia artificial

2.2.2. Redes neuronales

Las redes neuronales son métodos de inteligencia artificial que enseñan a las computadoras a procesar datos de manera inspirada en la forma en que lo hace el cerebro humano. Las redes neuronales pueden aprender y modelar relaciones complejas entre los datos de entrada y salida que no son lineales. Una red neuronal básica consta de neuronas artificiales interconectadas en tres capas:

- Capa de entrada: Los datos entran en la red neuronal artificial desde la capa de entrada. Los nodos de entrada procesan los datos, los analizan o los clasifican y lo pasan a la siguiente capa.
- Capas ocultas: Las capas ocultas toman su entrada de la capa de entrada o de otras capas ocultas. Las redes neuronales pueden tener una gran cantidad de capas ocultas. Cada capa oculta analiza la salida de la capa anterior, la procesa aún más y la pasa a

la siguiente capa.

- Capa de salida: La capa de salida proporciona el resultado final de todo el procesamiento de datos que realiza la red neuronal. Puede tener uno o varios nodos de salida dependiendo del resultado que se quiera.

Las redes neuronales tienen varias capas ocultas con millones de neuronas artificiales conectadas entre sí [3]. Los pesos representan las conexiones entre los nodos. Si bien las redes neuronales requieren un entrenamiento más extenso en comparación con otros métodos de aprendizaje automático, su capacidad para aprender representaciones no lineales de datos los convierte en una herramienta poderosa para resolver problemas complejos como la recomendación de ropa y la compatibilidad de atuendos.

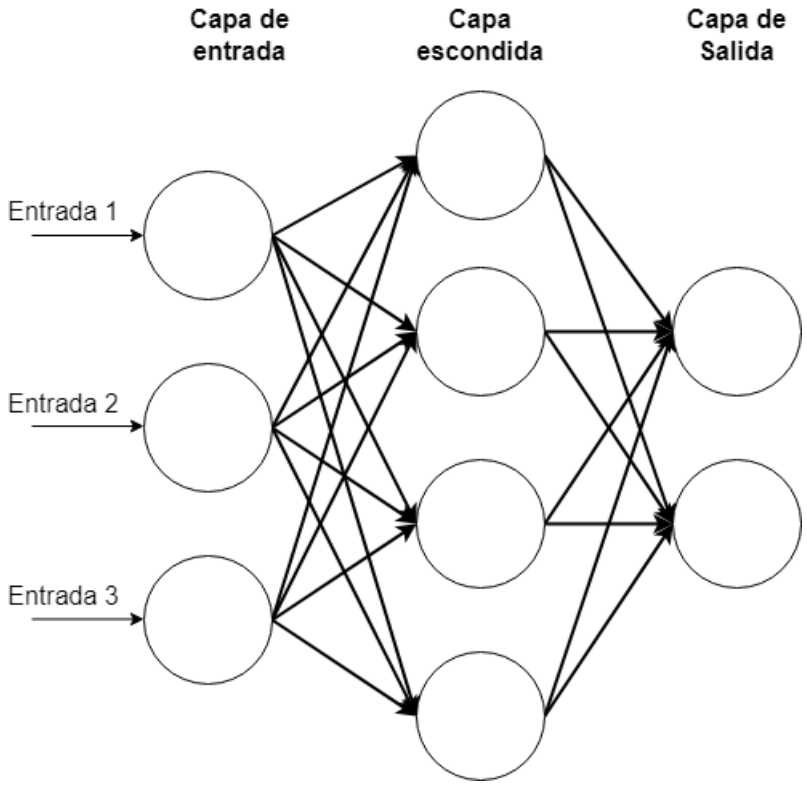

Figura 4: Ejemplo de una red neuronal que está compuesta por una capa de entrada con 3 neuronas, una capa escondida con 4 neuronas y una capa de salida con 2 neuronas.

2.3. Arquitecturas de Modelos

A continuación se revisan las arquitecturas importantes para entender el trabajo:

2.3.1. Perceptrón

El perceptrón es un algoritmo para el aprendizaje supervisado de clasificadores binarios. Un clasificador binario es una función que puede decidir si una entrada, representada por

un vector de números, pertenece o no a una clase específica. Es un tipo de clasificador lineal, es decir, un algoritmo de clasificación que realiza sus predicciones en función de una función de predictor lineal que combina un conjunto de pesos con el vector de características. El perceptrón toma un conjunto de entradas, realiza una combinación lineal ponderada de las entradas utilizando pesos asignados a cada una de ellas, y luego aplica una función de activación para producir una salida.

La idea es usar diferentes pesos que representen la importancia de cada entrada, y que si la suma ponderada de las entradas es mayor a un límite entonces se retorna 1, sino, se retorna 0. El perceptrón es la base de las redes neuronales.

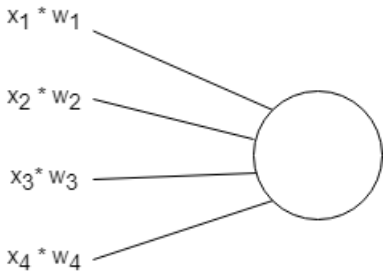

Figura 5: Arquitectura de un perceptrón con una sola capa

El perceptrón se puede generalizar para múltiples clases, en lo que se llama perceptrón multicapa. En esta red, cada unidad en una capa se conecta con todas las unidades en la capa anterior y en la siguiente capa mediante conexiones ponderadas. El término "multicapa" se utiliza para referirse a que hay mas de una capa oculta entre la capa entrada y la capa de salida. Esta configuración le permite a la red aprender relaciones no lineales entre variables y extraer características más relevantes de los datos.

2.3.2. Redes Neuronales Convolucionales

El modelo de redes neuronales convolucionales, también conocido como CNN, es utilizado para el análisis de imágenes, clasificar elementos visuales y realizar tareas de visión artificial. Se destacan de otras redes neuronales por su rendimiento superior para entradas de señales de imagen, voz o audio. Tienen tres tipos principales de capas, capa convolucional, capa de agrupación y capa completamente conectada.

- Capa convolucional: Es el bloque de construcción central de la CNN y son las encargadas de aplicar la convolución a las imágenes de entrada. Estas capas se definen por el número de filtros, también llamado *kernels*, que corresponden al número de matrices por las que se van a convolucionar las imágenes de entrada, el tamaño de los filtros y el *stride*. La entrada consta de 3 canales (R, G, B), lo que implica que la imagen de entrada se define mediante tres matrices, una para cada canal. La capa convolucional aplica la convolución de manera independiente para cada canal. Esto genera un resultado para cada canal, que se suma para obtener una única matriz bidimensional conocida como mapa de activaciones. El valor de *stride* determina cuántas posiciones, ya sea columnas

o filas, avanza el *kernel* en cada paso de convolución. Trás una capa de convolución, una CNN aplica una transformación ReLU al mapa de activaciones, lo que introduce la no linealidad al modelo.

- Capa de agrupación: Reduce el tamaño de los mapas de activaciones, lo que reduce el número de parámetros en la entrada. Esto ayuda a reducir la complejidad, mejorar la eficiencia y limitar el riesgo de sobre-ajuste.
- Capa completamente conectada: En la capa completamente conectada, cada nodo de salida se conecta directamente a un nodo de la capa anterior. Esta capa realiza la tarea de clasificación en base a las características extraídas a través de las capas anteriores y sus diferentes filtros. Esta capa permite que la red aprenda relaciones más complejas entre las características y las utilice para hacer predicciones finales mas precisas.

La información se obtiene de la siguiente fuente [11].

2.3.3. Red Neuronal Residual

Se han introducido redes neuronales más profundas mediante la adición de capas con el propósito de abordar tareas más complejas y mejorar la precisión en la tareas. Sin embargo, al incrementar el número de capas en la red neuronal, se vuelve más complejo su entrenamiento y la precisión empieza a estancarse y disminuir. Este fenómeno se conoce como el "problema del desvanecimiento del gradiente". Introducido por Kaiming et al. [8], ResNet surge como una solución a esta problemática.

El "problema del desvanecimiento del gradiente" en redes neuronales profundas hace referencia a la dificultad que surge durante el proceso de entrenamiento, donde el gradiente usado para actualizar los pesos de la red disminuye gradualmente al propagarse desde las capas de salida hacia las capas de entrada. Durante el entrenamiento, el objetivo principal consiste en minimizar una función de pérdida mediante el ajuste de pesos y parámetros de la red. Para hacer esto, se utiliza el algoritmo de retro-propagación, que calcula los gradientes al propagar el error desde la capa de salida hasta la capa de entrada. Sin embargo, este fenómeno del desvanecimiento del gradiente puede dar lugar a una convergencia lenta del entrenamiento, la posibilidad de quedar atrapado en mínimos locales y malas representaciones [5].

Una solución al problema la ofrece el modelo ResNet con la introducción de los "bloques residuales". En esta estructura, se usa una técnica llamada *skip connections* o conexiones de salto, que crea conexiones directas entre ciertas capas y otras, saltando capas intermedias en el proceso. Esta conexión alternativa da origen a los mencionados "bloques residuales". En la construcción de la red neuronal residual, múltiples bloques residuales se combinan estratégicamente. Una gran ventaja de incorporar estas conexiones de salto está en que si una capa específica está impactando negativamente el rendimiento general de la arquitectura, dicha capa puede ser ignorada gracias a la aplicación de técnicas de regularización.

En este trabajo se utiliza una variante específica del modelo ResNet, denominada ResNet-18, desarrollada por *Microsoft Research* [8]. Esta variante tiene 18 capas de profundidad y está diseñada específicamente para tareas de clasificación de imágenes. En la implementación

que se utiliza en este trabajo, se puede cargar una versión pre-entrenada de la red entrenada en más de un millón de imágenes desde la base de datos de *ImageNet*. El tamaño de las imágenes de entrada es de $224 \times 224 \times 3$.

2.3.4. Red Recurrente

Una red neuronal recurrente (RNN) es un tipo de red neuronal que se utiliza para procesar datos secuenciales. Estas redes son comúnmente utilizadas en problemas temporales, como traducción de idiomas, procesamiento de lenguaje natural (NLP), reconocimiento de voz, entre otros. Se destacan por su capacidad de "memoria", ya que pueden utilizar información de entradas anteriores para influir en las entradas y salidas actuales. A diferencia de las redes neuronales tradicionales, que asumen que la entrada y la salida son independientes entre sí, las redes recurrentes toman en cuenta la información de los elementos anteriores dentro de la secuencia para generar la salida actual [13].

Por ejemplo, en el caso del procesamiento de lenguaje natural para que una frase tenga sentido, las redes recurrentes necesitan considerar la posición de cada palabra en la frase y utilizar esa información para predecir la siguiente palabra en la secuencia.

La característica más importante de las redes recurrentes son sus estados escondidos o *Hidden States*, que recuerdan alguna información sobre una secuencia. El estado también es referido como *Memory State* porque recuerda la entrada anterior de la red. Usa los mismos parámetros para cada entrada y realiza la misma tarea en todas las entradas o entradas ocultas para producir el resultado. Esto reduce la complejidad de la red.

Se explica el funcionamiento de la arquitectura usando la Figura 6 de una red neuronal recurrente básica. Se considera una entrada como una secuencia de ítem $x = x_1, x_2, ..., x_i, ..., x_n$. Cada entrada x_i se entrega a la red junto al estado de la red anterior. Este estado de la red, que se muestra como h_i y denominado *hidden state*, se obtiene de haber alimentado los ítem de la secuencia pasada a la red y funciona como una especie de memoria.

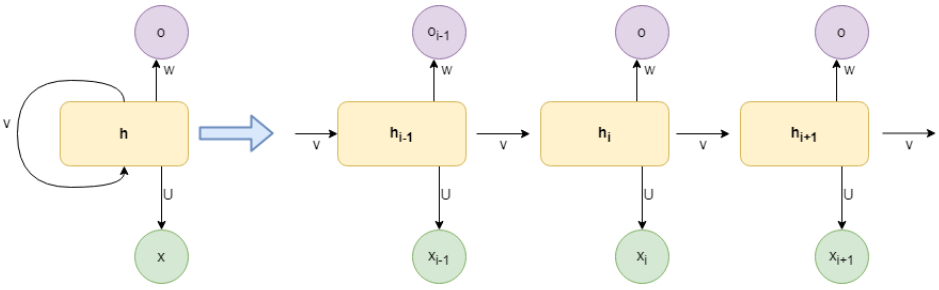

Figura 6: Arquitectura de la Red Neuronal Recurrente

2.3.5. Long Short-Term Memory Network

Long Short-Term Memory [9], o también llamado LSTMs, es una red neuronal secuencial que permite que la información persista. Las redes LSTMs resuelven el problema de desvane-

cimiento del gradiente al estar diseñadas para evitar problemas de dependencia a largo plazo. En la Figura 7 se muestra la estructura de la red LSTMs.

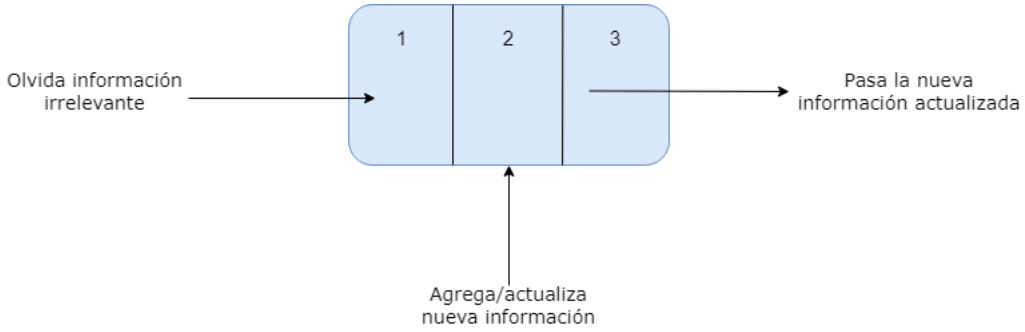

Figura 7: Arquitectura de la Red LSTM

El modelo funciona de manera similar a una unidad de RNN. La arquitectura consiste en tres compuertas: *Forget Gate*, *Input Gate* y *Output Gate*, y el funcionamiento de una unidad de LSTM es de forma secuencial.

1. *Forget Gate*: La primera compuerta decide si la información del *timestamp* previo va a ser recordada o es irrelevante y se puede olvidar.
2. *Input Gate*: En la segunda compuerta, la celda intenta aprender información nueva de la entrada.
3. *Output Gate*: La celda pasa la información actualizada a la próxima iteración.

Este ciclo completo se dice que dura una iteración completa. Es importante notar que una red LSTM tiene un estado escondido H_t y estado de celda C_t de la iteración t. H_t corresponde a la memoria a corto plazo y C_t la memoria a largo plazo.

Figura 8: Celda de la Red LSTM

2.3.6. Bidirectional Long Short Term Memory

Una red Bi-LSTM (*Bidirectional Long Short Term Memory*) es un tipo de arquitectura de red neuronal recurrente (RNN) que procesa la información de entrada que posee un orden para ambos lados, hacia al final y del final hasta el primero. Esto permite al modelo considerar el contexto futuro dada una entrada de un item de una secuencia, a diferencia del modelo LSTMs.

El modelo Bi-LSTMs consiste en dos redes LSTM, una que procesa la secuencia de entrada en el sentido hacia adelante y la otra hacia atrás. Esto le permite al modelo acceder a la información de las iteraciones pasadas y futuras al mismo tiempo. Esto permite al modelo obtener mejores resultados en tareas para comprender la secuencia de entradas.

2.3.7. Redes Siamesas

Una red neuronal siamesa (SNN) es una clase de arquitectura de redes neuronales que contienen dos o mas subredes idénticas, donde se define que son idénticas si tienen la misma arquitectura con con los mismos parámetros y pesos. La actualización de parámetros se refleja en ambas subredes y se utiliza para encontrar similitudes entre las entradas mediante la comparación de los vectores de características.

Las redes neuronales tradicionales aprenden a predecir múltiples clases predefinidas, lo que plantea un problema cuando se necesitan agregar o eliminar nuevas clases de datos. En este caso, se tiene que actualizar la red neuronal y volver a entrenar en todo el conjunto de datos. Por el otro lado, como las redes neuronales siamesas usan una función de similitud, se puede entrenar la SNN para determinar si dos imágenes son iguales. Eso le permite a la red clasificar nuevas clases de datos sin tener que volver a entrenar toda la red. [18]

2.4. Funciones de pérdida

Como se explicó anteriormente, en este trabajo se utiliza principalmente la función de pérdida *Ranking Loss*, cuyo objetivo es predecir las distancias relativas entre entradas, una tarea conocida como *metric learning*.

Lo más importante de las funciones de *Ranking Loss* es que se necesita un puntaje de similitud entre los datos para utilizarlas. Para emplear esta función de pérdida, primero se extraen características de los datos que se van a comparar y se obtienen representaciones para cada ítem. Luego, se define una función de métrica para medir la similitud entre las representaciones, por ejemplo, la distancia euclidiana. Finalmente, se entrenan los extractores de características para producir representaciones similares para ambas entradas en el caso de que las entradas sean similares, o representaciones distantes para dos entradas en el caso de que sean diferentes. Solo importa la distancia entre las representaciones, no el valor de ellas. Esta metodología de entrenamiento ha demostrado producir representaciones poderosas [6].

Se destacan dos tipos de *Ranking Losses* para dos configuraciones distintas, cuando se usan pares de datos y tripletas de datos de entrenamiento.

2.4.1. Pairwise Ranking Loss

En esta configuración, se utilizan ejemplos de pares positivos y negativos. Los datos positivos están compuestos por un ejemplo de ancla x_a y un ejemplo positivo x_p. Los datos negativos están compuestos por un ejemplo de ancla x_a y un ejemplo negativo x_n, que es disimilar a x_a en esa métrica.

El objetivo es aprender una representación con una distancia pequeña d entre ellos para

los pares positivos y una distancia mayor que un margen m para pares negativos:

$$\begin{aligned} L &= d(r_a, r_p) && \text{si es un par positivo} \\ L &= \text{máx}(0, m - d(r_a, r_n)) && \text{si es un par negativo} \end{aligned}$$

Para ejemplos positivos, la pérdida será 0 solo cuando la red produce representaciones para ambos elementos en el par sin distancia entre ellas, y la pérdida aumentará con esa distancia.

Para ejemplos negativos, la pérdida será 0 cuando la distancia entre las representaciones de los dos elementos del par sea mayor al margen m. Pero cuando esa distancia no sea mayor que m, la pérdida será positiva y los parámetros de la red se actualizarán para producir una representación más distante de esos dos elementos. El valor de la pérdida será a lo sumo m cuando la distancia entre r_a y r_n sea 0. La función del margen es que, cuando las representaciones producidas para un par negativo sean lo suficientemente distantes, no se pierda esfuerzo en ampliar esa distancia.

2.4.2. Triplet Ranking Loss

Esta configuración supera a la anterior mediante el uso de tripletas de muestras de datos de entrenamiento en lugar de pares. Las tripletas están conformadas por una muestra ancla x_a, una muestra positiva x_p y una muestra negativa x_n. El objetivo es que la distancia entre el ejemplo de ancla y el ejemplo negativo $d(r_a, r_n)$ sea mayor (y mayor a un margen m) que la distancia entre un ejemplo de ancla y un ejemplo positivo $d(r_a, r_p)$. Se puede describir de la siguiente forma:

$$L(r_a, r_p, r_n) = \text{máx}(0, m + d(r_a, r_p) - d(r_a, r_n))$$

Una gran cantidad de modelos de recomendación utilizan esta función de pérdida.

2.4.3. Selección de negativos

Una decisión importante en el entrenamiento con *Triplet Ranking Loss* es la selección de ejemplos negativos, también denominado *triplet mining*. La estrategia que se elija tendrá un alto impacto en la eficiencia del entrenamiento y en el rendimiento final.

2.5. Optimizadores

A continuación se revisan los optimizadores más importantes para el trabajo:

2.5.1. Método de descenso de gradiente

El método del descenso de gradiente es un algoritmo de optimización que se fundamenta en la noción de actualizar de manera iterativa los parámetros en dirección opuesta al gradiente de la función de pérdida, con el propósito de identificar un valor mínimo. Este optimizador modifica consistentemente los valores de los parámetros en busca de una convergencia hacia un mínimo local. Su denominación proviene de la estrategia de seguir la dirección del gradiente descendente de la función de pérdida, con el fin de ubicar el mínimo, ya sea local o global.

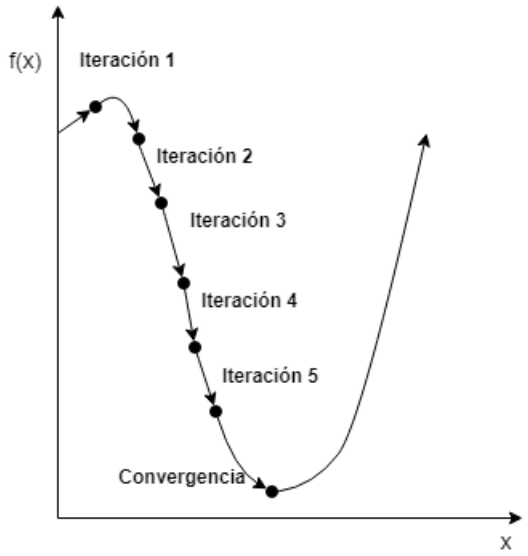

Figura 9: Optimizador de Descenso de gradiente

2.5.2. Método de descenso de gradiente con momento

Stochastic Gradient Descent with Momentum o, simplemente, *Momentum*, es una variación del algoritmo de descenso de gradiente estocástico (SGD) que utiliza una técnica de suavizado para evitar oscilaciones excesivas en la dirección del gradiente y aumentar la velocidad de convergencia: En lugar de actualizar los parámetros utilizando solo la información del gradiente actual (tras evaluar n muestras), el algoritmo de *Momentum* utiliza un promedio ponderado del gradiente actual y de los gradientes anteriores para calcular la dirección de actualización (es decir, el gradiente a considerar para la actualización de los parámetros).

Este algoritmo requiere dos parámetros: el coeficiente de aprendizaje que ya conocemos (*learning rate*) y el llamado coeficiente de momento. El coeficiente de momento controla la influencia de los gradientes anteriores en la dirección de actualización. En función de los datos, un coeficiente de momento alto puede ayudar a superar los mínimos locales en la función de error y llevar a una convergencia más rápida.

La actualización de los pesos en el algoritmo de Momento se realiza, por lo tanto, de la siguiente manera:

1. Se evalúa el gradiente actual tras considerar n muestras.
2. Se calcula la dirección de actualización como una combinación del gradiente actual y los gradientes anteriores ponderados usando el coeficiente de momento.
3. Se actualizan los parámetros del modelo utilizando la dirección de actualización y el coeficiente de aprendizaje.

El algoritmo de Momento se puede implementar de diferentes maneras, como utilizando un promedio ponderado simple de los gradientes anteriores o utilizando un promedio ponderado exponencial.

2.5.3. RMSprop

RMSProp o *Root Mean Square Propagation* propone que, en lugar de mantener un acumulado de los gradientes, se utiliza el concepto de "ventana" para considerar solo los gradientes más recientes, aplicándose a éstos una media exponencial ponderada para suavizar los cambios aplicados a los parámetros. Este enfoque puede ayudar a evitar que las tasas de aprendizaje se vuelvan excesivamente pequeñas.

2.5.4. Adam

El optimizador de Adam, o *Adapative Moment Estimation*, es un método de optimización que combina las ventajas de los algoritmos RMSprop y Momentum para mejorar el proceso de aprendizaje de un modelo. Al igual que en el método de Momento, Adam utiliza una estimación del momento y de la magnitud de los gradientes anteriores para actualizar los parámetros del modelo en cada iteración. Sin embargo, en vez de utilizar una tasa de aprendizaje constante para todos los parámetros, Adam adapta la tasa de aprendizaje de cada parámetro individualmente en función de su estimación de momento y de la magnitud del gradiente. Esto permite que el modelo consiga una mejor precisión en la predicción. Este es el principal optimizador que se trabaja en los modelos.

2.6. *Backbone*

En el contexto de aprendizaje de máquinas, el término *backbone* se refiere a la parte central de una arquitectura de red utilizada para la clasificación de imágenes. Es la estructura fundamental que permite a la red extraer características significativas de los datos de entrada [17]. En este trabajo, el *backbone* principal de los modelos a evaluar es la red *ResNet*, específicamente la variante *ResNet-18*, que consta de 18 capas y utiliza bloques residuales para facilitar el entrenamiento en redes más profundas.

2.7. Modelos Contrastivos

2.7.1. BYOL

BYOL [7] es una arquitectura auto-supervisada contrastiva cuyo objetivo es aprender representaciones de imágenes que puedan ser usadas para tareas posteriores. BYOL consiste de dos redes neuronales: las ramas *online* y *target* que interactúan una con la otra.

Los primeros enfoques abordan el problema de predicción directamente en el espacio de representaciones. Por ejemplo, se busca que la representación de una vista aumentada de una imagen pueda predecir la representación de otra vista aumentada de la misma imagen. Sin embargo, esta aproximación puede llevar a representaciones colapsadas, como una representación constante en todas las vistas que siempre se predice a sí misma. Los métodos contrastivos resuelven este problema reformulando la predicción como un ejercicio de discriminación. A partir de una representación de una vista aumentada, se entrena al modelo para distinguir entre la representación de otra vista aumentada de la misma imagen y las representaciones de vistas aumentadas de imágenes diferentes, evitando así representaciones colapsadas en la mayoría de los casos.

Esto conllevaba generar ejemplos negativos para cada caso, y como se explica anteriormente, la elección de ejemplos negativos es una tarea compleja, requiriendo usar diversas estrategias como tamaños de lotes grandes, bancos de memoria u otros métodos para obtener las parejas negativas. El rendimiento dependía también de la elección de las transformaciones a aplicar a las imágenes.

En contraste, BYOL no discrimina entre representaciones de vistas aumentadas de diferentes imágenes. Para prevenir el colapso, utiliza una representación conocida como ”objetivo” para entrenar una nueva representación llamada ”en línea”, mediante la predicción de la representación ”objetivo”. Esto permite construir una secuencia de representaciones de mayor calidad al iterar este proceso, utilizando redes en línea posteriores como nuevos objetivos para entrenamientos adicionales. La red objetivo se actualiza con un promedio lento de la red en línea.

Es importante destacar que se aplican transformaciones a las vistas aumentadas de una imagen. A continuación se muestra la arquitectura de BYOL:

Se puede destacar la red *online*, la cual está definida por una serie de pesos y compuesta por tres etapas: un codificador que equivale al *backbone* de la red, un proyector que corresponde a un MLP (Perceptrón Multicapa) y un predictor que utiliza la misma arquitectura que el proyector. Por otro lado, la red *target* presenta una estructura similar, pero sin el predictor y utilizando un conjunto de pesos diferente. En este contexto, la red de proyección MLP lleva la representación del *backbone* a un espacio de representación de menor dimensionalidad.

Dada una imagen x, el procedimiento para computar la pérdida es el siguiente:

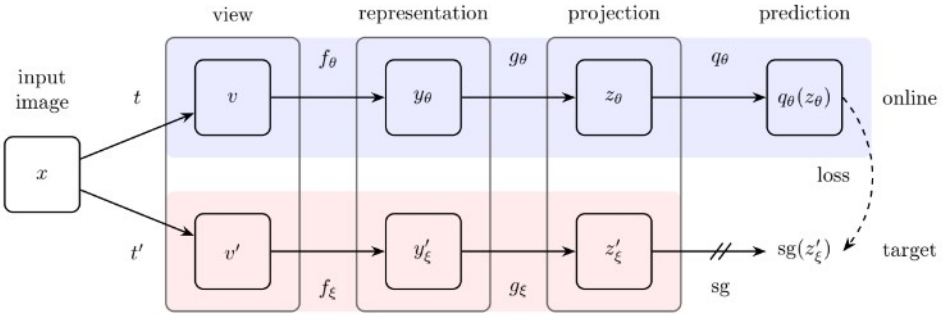

Figura 10: Arquitectura del modelo BYOL [7].

- Se aplican las transformaciones t y t' a la imagen de entrada x en las ramas *online* y *target* respectivamente para obtener así las vistas v y v'.
- Se computan las representaciones de las vistas de las imágenes en las redes *online* y *target* con las funciones f_θ y f_ε y obtener así y_θ y y'_ε. Estas funciones corresponden al *backbone* del modelo.
- Se obtienen las proyecciones de las imágenes aplicando las funciones g_θ y g_ε respectivamente. Las proyecciones corresponden a las MLP que llevan las representaciones a un espacio de menor dimensionalidad.
- Se calcula una predicción de $q_\theta(z_\theta)$ de la proyección de la rama *target* z'_ε, que también corresponde a una red MLP.
- Por último, se define una función de error para medir la similitud entre las vistas de la imagen, que corresponde a la distancia euclideana.

Ya entrenado el modelo, se obtienen las representaciones de las imágenes de la capa *online*. Es decir, se producen todos los y_θ.

Las transformaciones que aplica BYOL a ambas vistas de la imagen original corresponden a:

- Recorte aleatorio: se selecciona un parche aleatorio de la imagen y se redimensiona al tamaño objetivo de 224×224.
- Inversión izquierda-derecha.
- *Jittering* de color: Los valores de brillo, contraste, saturación y tono de la imagen experimentan desplazamientos mediante una desviación aleatoria uniforme, la cual se aplica a todos los píxeles de la imagen. El orden en que se ejecutan dichos desplazamientos se selecciona aleatoriamente para cada fragmento
- Disminución de color: Se ofrece la opción de realizar una conversión a escala de grises. En caso de aplicarse, la intensidad de salida para un píxel (r, g, b) corresponde a su componente de luminancia.
- Desenfoque Gaussiano: Para una imagen de 224×224, se emplea un núcleo gaussiano cuadrado de tamaño 23×23, con una desviación estándar obtenida de un muestreo uniforme en el intervalo [0.1, 2.0].

Cada una de estas transformaciones se aplica con una probabilidad predefinida.

2.7.2. Simple Siamese

Simple Siamese [20], como su nombre sugiere, es un tipo de red neuronal siamesa auto-supervisada para obtener representaciones basadas en imágenes. Utiliza una función de pérdida contrastiva con ejemplos positivos solamente. Similar a BYOL, y a diferencia de otros modelos contrastivos, logra generar representaciones de alta calidad sin necesidad de ejemplos negativos, lotes de datos extensos o codificadores de momentum.

La estructura de la red consta de dos arquitecturas idénticas que corresponden a los codificadores f, los cuales incluyen un *backbone* y una proyección MLP, además de un predictor MLP h. En esta red, el *backbone* está compuesto por una ResNet preentrenada en un conjunto amplio de datos. Esta elección permite extraer características generales que luego se pueden transferir a tareas específicas con conjuntos de datos más reducidos.

La proyección MLP, al igual que en el modelo BYOL, se refiere a una red Perceptrón multicapa y lleva la vectorización a un espacio de menor dimensionalidad. Por último, el predictor MLP tiene la misma arquitectura que la proyección MLP, pero su objetivo es predecir resultados o salidas a partir de una entrada.

La arquitectura de la red se presenta a continuación:

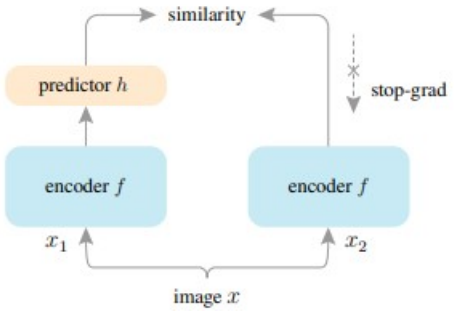

Figura 11: Arquitectura del modelo SimSiam [20]

El proceso para una imagen de entrada es el siguiente:

1. Se suministran dos vistas de la imagen, x_1 y x_2, o se aplican dos transformaciones a una imagen x, a un mismo codificador f, que consiste en el *backbone* y una proyección MLP.
2. Se aplica una predicción MLP h a una de las representaciones de la imagen, mientras que se aplica la operación *stop-gradient* a la otra. Esta operación detiene la propagación del gradiente a través de ciertas variables o capas de la red durante la retropropagación. Esto previene que los gradientes retrocedan y alteren esas partes del modelo durante el proceso de ajuste fino.

3. El modelo maximiza la similitud entre ambas representaciones usando una función de pérdida similitud coseno negativo.

El modelo *Simple Siamese* presenta una ventaja clave en comparación con BYOL, especialmente en términos de simplicidad y eficiencia en la estructura y entrenamiento. *Simple Siamese* no utiliza la estrategia de actualización de la red de momentum que BYOL utiliza para mejorar las representaciones, lo que simplifica significativamente el entrenamiento de la red. Es importante destacar que en la implementación original de *Simple Siamese* se aplican la misma serie de transformaciones que para BYOL.

2.8. *Django*

Django es un robusto *framework* para agilizar y simplificar el proceso de desarrollo de aplicaciones web. Escrito en el lenguaje de programación *Python*, *Django* ofrece una amplia gama de herramientas y características que permiten a los desarrolladores crear aplicaciones web de manera eficiente y efectiva.

Un *framework* web proporciona una estructura predefinida y componentes reutilizables para acelerar el desarrollo de aplicaciones. *Django* ofrece herramientas para manejar: la gestión de la autenticación de usuarios, la creación de un panel de administración completo para la aplicación, la implementación de formularios y la capacidad de subir archivos, entre otras cosas. Esto permite a los desarrolladores concentrarse en la lógica específica de la aplicación en lugar de escribir repetitivamente código básico.

En este trabajo se utiliza *Django* como framework de desarrollo web debido a las siguientes ventajas:

1. Proporciona una estructura sólida para el desarrollo de aplicaciones personalizadas. Para este trabajo se utiliza la base con la serie de herramientas que ofrece *Django* para crear una aplicación específica para que muestra las recomendaciones de ropa.
2. Permite crear interfaces web con una mínima cantidad de código. El *framework* se encarga de gran parte del trabajo subyacente, lo que reduce la necesidad de escribir código repetitivo. Esto permite enfocarse en la implementación de lógica de la aplicación y en las características específicas del trabajo.

Capítulo 3

Estado del Arte

Existen diversos trabajos que han tratado el problema de *fashion compatibility* basado en imágenes usando modelos de aprendizaje. A continuación se evalúan los modelos más relevantes bajo estas condiciones:

3.1. Modelos supervisados

Los modelos supervisados mencionados requieren conjuntos de tripletes que contengan ejemplos tanto positivos como negativos de las prendas, junto con sus respectivas categorías.

3.1.1. Learning Fashion Compatibility with Bidirectional LSTMs

La tarea de *fashion compatibility* se aborda inicialmente como un problema secuencial utilizando LSTM bidireccionales [21]. La idea general del modelo es predecir el siguiente ítem compatible entre las prendas que conforman una vestimenta.

Una vestimenta se representa como una secuencia de items de moda. En este modelo, se aprende un espacio de entradas multimodales de usuarios, que incluye tanto texto describiendo atributos como imágenes. Para lograrlo, se emplea una red Bi-LSTM que se entrena para predecir secuencialmente el siguiente ítem condicionado a los items anteriores que se han visto en ambas direcciones.

Además, el modelo aprende vectorizaciones semánticas visuales al proyectar la imagen y sus descripciones a un espacio que incorpora información de sus atributos y la categoría. Esto no solo proporciona una representación semántica de la entrada para regularizar el entrenamiento del Bi-LSTM, sino que también permite la generación de una vestimenta con entradas multimodales de los usuarios. El modelo aprende de manera conjunta tanto la compatibilidad de moda como la incrustación visual-semántica, incorporando así información semántica valiosa en el proceso de entrenamiento del modelo Bi-LSTMs.

Las ventajas de este modelo son:

- Puede modelar dependencias temporales de largo alcance a través de pasos de tiempo.
- Se entrena explícitamente para capturar las relaciones de compatibilidad, así como el estilo general de todo el atuendo.
- Modela la compatibilidad de una vestimenta entera, no solo de pares de prendas de

ropa.

- Realiza las tareas de recuperación y compatibilización de ropa: esto se refiere a que puede entregar una vestimenta compatible con una prenda, y computar un puntaje de compatibilidad si se entrega como argumento una vestimenta entera respectivamente.

Las desventajas del modelo:

- Las imágenes deben incluir una descripción y la categoría a la que pertenecen. Esto aumenta los requisitos de datos y complejidad para tener etiquetas descriptivas asociadas a las imágenes.

3.1.2. Learning visual clothing style with heterogeneous dyadic co-ocurrences

El modelo propuesto por Veit et al. [19] tiene como principal objetivo aprender una transformación de características de imágenes de items en un espacio latente de alta dimensionalidad llamado *style space*, mediante el uso de una arquitectura convolucional siamesa. A diferencia de trabajos anteriores, este enfoque no requiere predefinir clases específicas de estilo, sino que aprende un espacio continuo de alta dimensionalidad.

El *style space* es espacio de representaciones continua de alta dimensionalidad, donde los items compatibles se encuentran cercanos entre sí, mientras que aquellos que no son compatibles están más alejados. Los ejemplos de entrenamiento consisten en pares de items que son compatibles o no compatibles, es decir, ejemplos positivos o negativos, respectivamente.

El funcionamiento del modelo es el siguiente:

1. La entrada consiste en imágenes de items de ropa, etiquetas de las categorías y los enlaces entre los items.
2. Se muestrean ejemplos de entrenamiento a partir de los datos de entrada, que son pares de items co-ocurrentes de distintas categorías y que ocurren con frecuencia.
3. Se utiliza una red Siamesa CNN para aprender una transformación de características desde el espacio de imágenes al espacio latente.
4. Se generan paquetes de ítems compatibles al hacer una consulta al espacio latente y obtener el *Nearest Neighbor* de cada categoría al ítem de la consulta.

Las principales ventajas de este modelo son:

- Generaliza bien para categorías de ropa no antes vistas, incluso con un conjunto de datos pequeño.
- Crea paquetes de pares de prendas, que indican la compatibilidad entre items de distintas categorías. En lugar de enfocarse en la similitud, el modelo aprende la compatibilidad entre prendas.
- Aprende nociones de estilo de forma independiente, por lo que no es necesario predefinir

las categorías de estilo previamente.

Por otro lado, las principales desventajas del modelo son:

- El espacio es unidimensional y puede que no sea capaz de capturar todas las relaciones de compatibilidad complejas del estilo. Esto podría llevar a problemas como *improper triangles* o falta de transitividad en las relaciones de compatibilidad.
- La definición de compatibilidad basada en productos frecuentemente comprados juntos puede ser errónea y no garantiza que dos productos sean realmente compatibles. El hecho de que dos items sean adquiridos juntos no necesariamente significa que se complementen en una vestimenta.

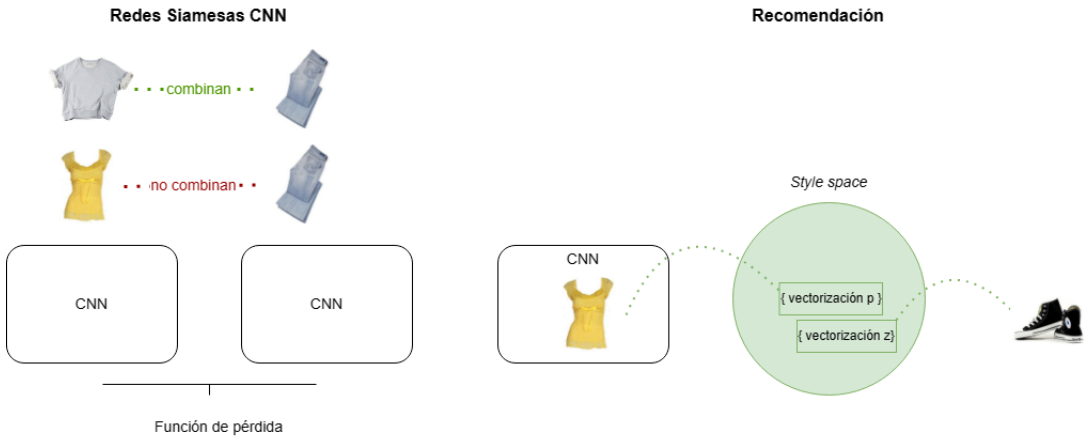

Figura 12: Funcionamiento de la red propuesta por Viet et al. [19]

3.1.3. Learning Type-Aware Embeddings for Fashion Compatibility

En el modelo propuesto por Vasileva et al. [10], las imágenes se representan respetando su categoría y aprendiendo conjuntamente nociones de similitud y compatibilidad. Se comienza representando cada imagen en un espacio general de vectores que se utiliza para medir la similitud entre items. Además, se aprende una proyección que mapea las vectorizaciones generales a un espacio secundario de vectorizaciones que proporciona el puntaje de compatibilidad entre dos categorías de prendas. A diferencia del trabajo previo que utiliza un único espacio de vectores para obtener un puntaje de compatibilidad, este enfoque utiliza una proyección y subespacio distintos para cada par de tipos de prendas.

Las ventajas de este modelo son las siguientes:

- Aprende representaciones que pueden encapsulan tanto nociones de similitud como relaciones de compatibilidad entre prendas.
- Obtiene resultados comparables con el estado del arte en la tarea de *fashion compatibility*.

Las principales desventajas son las siguientes:

- Los subespacios de vectores están predefinidos, lo que implica que las categorías de ropa deben configurarse de antemano. Por lo tanto, no puede generalizar para categorías no vistas previamente.
- Aprende un subespacio por cada par de categorías, lo que incrementa de forma cuadrática con la cantidad de categorías, lo que puede ser computacionalmente costoso.

3.1.4. Learning Similarity Conditions Without Explicit Supervision

En un trabajo más reciente [15], se presenta un modelo que aborda la tarea aprendiendo distintos subespacios de vectores, los cuales pueden generalizar eficientemente a categorías no vistas previamente, sin requerir supervisión explícita o etiquetas previamente definidas. Este modelo logra un rendimiento superior en comparación con los enfoques anteriores.

La arquitectura del modelo se puede ver en la Figura 13. Para realizar la comparación entre dos imágenes, se emplea una red convolucional que extrae sus características, que se denominarán V_1 y V_2, en un espacio de vectores general. A continuación, estas características son procesadas por una rama de condiciones con el objetivo de aprender los subespacios relevantes para la tarea de compatibilidad. Después de aplicar diversas transformaciones, se obtienen representaciones finales que permiten identificar subespacios semánticos relevantes, donde la similitud entre ambas imágenes es comparada.

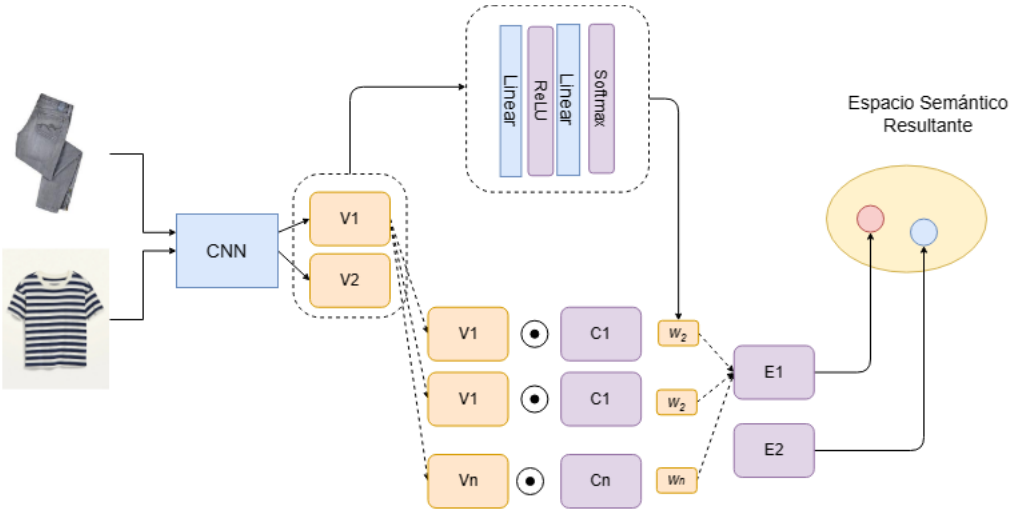

Figura 13: Arquitectura de la red SCE-Net [15]

El funcionamiento del modelo es el siguiente:

1. Las imágenes son proyectadas con un *backbone* a un espacio de características común que se denominará espacio de representaciones general. Dadas dos imágenes, las representaciones se denominarán V_1 y V_2.

2. Para determinar cuál subespacio semántico es relevante para la comparación, las características visuales se entregan a un *condition weight branch*, que es una red neuronal simple. Las condiciones de similitud son aplicadas usando producto punto a las características de las imágenes. Las máscaras de condiciones se pueden entender como un tipo de mecanismo de atención que realizan *dynamic assignment* de cada máscara de condición a los objetos que están siendo comparados. Se utiliza una rama de peso de condición para permitir que el modelo determine automáticamente qué conceptos aprender. La rama de peso de condición determina la relevancia de cada máscara de condición.

Las ventajas del modelo son las siguientes:

- Generaliza bien para categorías no antes vistas.
- Aprende pocos subespacios de dimensiones, en comparación del modelo anterior. Para resultados óptimos se necesitan aprender 5 subespacios.

Las desventajas principales son:

- No es eficiente para la tarea de recuperación de ropa.
- Si bien puede calcular el puntaje de compatibilidad de pares de prendas, resulta muy costoso en procesamiento y tiempo calcular el puntaje entre cada par de prendas y una con la que más combine.

3.1.5. Fashion Outfit Complementary Item Retrieval

En el modelo propuesto por Lin et al. [22], se aborda la cuestión de la compatibilidad en prendas de vestir, enfocándose en la recuperación en lugar de la predicción. La recuperación se refiere a la tarea de encontrar prendas de vestir similares o compatibles a partir de una consulta o referencia.

La arquitectura del modelo implementa una red de atención subespacial por categoría, un enfoque escalable para aprender atenciones subespaciales que permite búsquedas a gran escala, evitando comparaciones exhaustivas. A diferencia del modelo anterior, los factores condicionales de los subespacios son las categorías de los artículos de ropa. Esto posibilita extraer características de los artículos individuales mediante indexación y recuperación a gran escala.

El funcionamiento del modelo es el siguiente:

1. Se suministra una imagen de entrada a un *backbone* para extraer sus características.
2. Se aplica un conjunto de máscaras al vector de características de la imagen, generando múltiples representaciones de subespacios.
3. Se concatenan dos señales de categoría y se entregan a la subred para predecir los pesos del mecanismo de atención, el cual selecciona los vectores de subespacio adecuados para el cálculo del vector final.

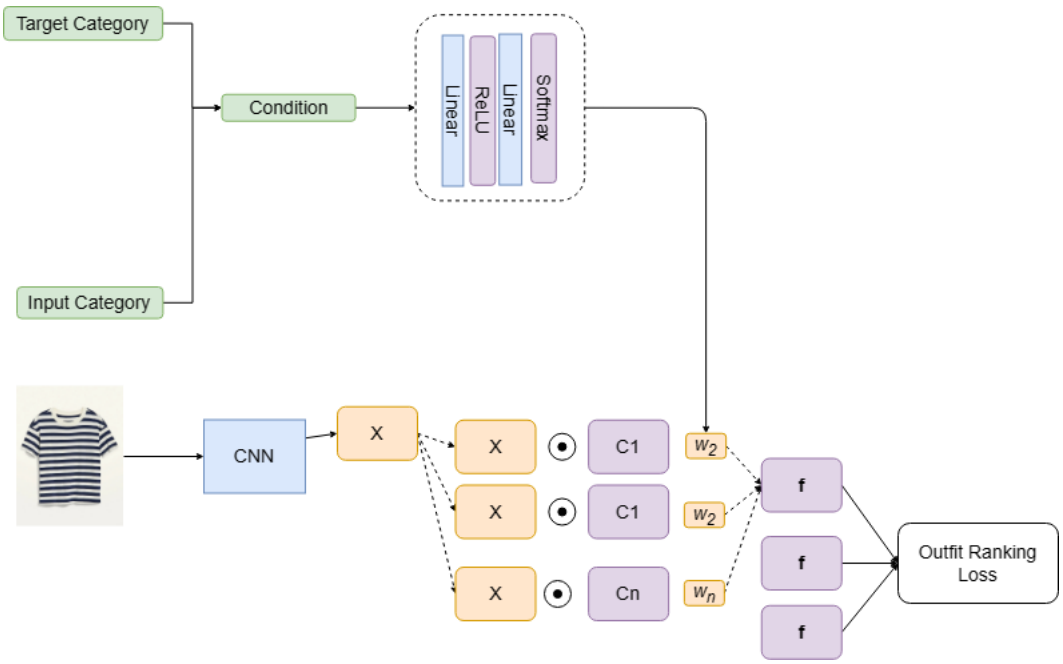

Figura 14: Arquitectura de la red CSA-Net [22]

4. La red se entrena con una función de pérdida basada en *Ranking Loss*, que considera todas las prendas.
5. Para la recuperación de prendas, se obtienen representaciones para cada categoría de destino.

Las ventajas del modelo son:

- Configurado para abordar la tarea de recuperación de prendas.
- Diseñado para calcular la puntuación de compatibilidad para conjuntos completos de vestimenta.
- Utiliza subespacios de estilo para capturar múltiples dimensiones de similitud, permitiendo identificar relaciones más complejas.

Las desventajas del modelo son:

- Requiere información de la categoría de la imagen de entrada y las categorías de destino, lo que implica que el conjunto de datos debe estar etiquetado para que el modelo funcione correctamente.

3.2. Modelos Auto-supervisados

Los modelos supervisados tienen ciertas dificultades, como el requerimiento de datos etiquetados y la dependencia de la elección de condiciones de similitud, las que se pueden superar utilizando modelos auto-supervisados. Estos modelos aprenden a vectorizar las imá-

genes de prendas en un espacio donde las prendas más cercanas entre sí son compatibles, sin necesidad de etiquetas para el entrenamiento. Esta aproximación puede ser más efectiva y escalable para resolver la tarea de compatibilidad de moda.

3.2.1. Self-supervised Visual Attribute Learning for Fashion Compatibility

El modelo presentado por Kim et al. [4] propone un modelo para aprender atributos visuales de colores y texturas invariantes a la forma sin necesidad de etiquetas durante el entrenamiento. El modelo aprende tres tareas auto-supervisadas diseñadas para capturar diferentes conceptos que se descuidan en los trabajos anteriores, según la necesidad de las tareas posteriores. El modelo está diseñado para aprender vectores de imágenes con color y patrones de textura similares cercanos los unos a otros. Las tareas corresponden a predecir color, textura y patrones en la textura usando parches cuadrados de las imágenes.

Las ventajas del modelo son:

- No necesita ejemplos negativos para ser entrenado.
- Aprende características invariante a la forma, dependiendo del color y la textura de las prendas de ropa.

Las desventajas son:

- Intenta descomponer la moda en distintos conceptos, por ejemplo color y textura. El problema de esto es que supone que prendas de ropa del mismo color combinan, lo que no necesariamente es cierto. Similitud no es lo mismo que compatibilidad.
- No se entrena con pares de ejemplos positivos, no aprende un sentido de compatibilidad inherente a la ropa, sino que hace suposiciones sobre qué prendas combinan.

3.2.2. Learning Fashion Compatibility from In-the-wild Images

El modelo presentado por Popli et al. [2] propone aprender representaciones para la predicción de compatibilidad a partir de imágenes de moda callejera en la naturaleza a través del aprendizaje auto-supervisado. El modelo funciona con el supuesto que las personas a menudo usan atuendos compatibles. Adicionalmente, para reducir la brecha de dominio entre las imágenes obtenidas de la calle y de catálogo durante la inferencia, se introduce una función de pérdida adversarial que minimiza la diferencia en la distribución de características entre los dos dominios. El modelo requiere características sin forma para medir la cercanía entre diferentes categorías, por lo que se recurre a representaciones basadas en parches.

El modelo se compone de un *framework* adversarial que incluye un generador, un discriminador y una red de vectorización. El generador, implementado como una red convolucional, utiliza parámetros para mapear la imagen de entrada a un espacio de representación. Por

otro lado, el discriminador es un perceptrón multicapa (MLP) de dos capas con un nodo de salida. Recibe como entrada las representaciones intermedias generadas por el generador y las clasifica en las categorías ”fuente” o ”objetivo”.

Las ventajas del modelo son las siguientes:

- Aprende parámetros del modelo invariante a la forma, dependiendo del color y la textura de las prendas de ropa.

Las desventajas corresponden a:

- Las vestimentas que utiliza como ejemplo provienen de casos reales de la calle. Considera que las prendas vestidas por la misma persona combinan pero esto no es necesariamente cierto.

Capítulo 4

Desarrollo

4.1. Definición de requisitos

En primer lugar, se definen las tareas y requisitos que deben cumplir los modelos de recomendación de ropa que serán comparados y evaluados, teniendo en cuenta su aplicación en un contexto de un comercio electrónico de moda. Utilizando una imagen de entrada $x_i^{(u)}$ que representa una prenda de la categoría u, las tareas a abordar son las siguientes:

- Recuperación de prendas de distinta categoría: Se busca mostrar una selección de prendas $x_j^{(v)}{}_{j=1}^{N}$ de categorías diferentes que sean compatibles con la prenda $x_i^{(u)}$.
- Recomendación de prendas similares: El objetivo es proporcionar una colección de prendas $x_j^{(u)}{}_{j=1}^{N}$ de la misma categoría que puedan ser intercambiables con $x_i^{(u)}$.

Es importante tener en cuenta ciertos requisitos que el modelo de recomendación debe cumplir:

- Entrada simplificada: El modelo debe aceptar únicamente una imagen de la prenda $x_i^{(u)}$, junto con la información de la categoría a la que pertenece (u) y la categoría de las prendas que se buscan (v).
- Eficiencia y rapidez: El modelo debe ser eficiente en el uso de recursos computacionales y capaz de recuperar las imágenes de prendas de la base de datos en poco tiempo durante su ejecución.
- Rendimiento comparativo: El objetivo es obtener resultados comparables con los modelos más avanzados y efectivos conocidos en el estado del arte.

De esta manera, se busca desarrollar un modelo de recomendación con el fin de mejorar la experiencia del usuario al ofrecerle opciones adecuadas de prendas.

4.2. Selección de modelos

Se evalúan los modelos mencionados en la sección de estado del arte de acuerdo a las tareas y requisitos.

1. Learning Fashion Compatibility with Bidirectional LSTMs: Este modelo no obtiene resultados en las métricas comparables al estado del arte. Además, el modelo entrega

un puntaje de compatibilidad para un conjunto de prendas que se encuentren en un orden secuencial. Esto no es lo que queremos para nuestro modelo de recomendación, en el que buscamos prendas compatibles con otra.

2. Learning visual clothing style with heterogeneous dyadic co-ocurrences: Se descarta porque su espacio de compatibilidad es unidimensional y no considera relaciones más complejas entre las categorías de las prendas. No posee resultados comparables con el estado del arte.
3. Learning Type-Aware Embeddings for Fashion Compatibility: Si bien el modelo recibe una entrada multimodal y utiliza las descripciones que acompañan a las imágenes para incorporar información semántica, se considerará como modelo base por su alto rendimiento en las métricas de evaluación.
4. Learning Similarity Conditions Without Explicit Supervision: Este modelo no es eficiente para la tarea de recomendación de prendas, no permite realizar indexación a gran escala. Por esta razón, no se considerara como modelo base.
5. Fashion Outfit Complementary Item Retrieval: Corresponde al modelo supervisado que obtiene los mejores resultados en las métricas definidas y permite realizar indexación y búsqueda de prendas a gran escala.
6. Self-supervised Visual Attribute Learning for Fashion Compatibility: No aprende un sentido de compatibilidad inherente a la ropa, sino que hace suposiciones sobre qué prendas combinan. Esto no es lo que queremos para nuestro modelo de recomendación, donde buscamos que aprenda el sentido de compatibilidad.
7. Self-supervised Visual Attribute Learning for Fashion Compatibility: Al igual que el modeelo S-VAL, hace suposiciones sobre qué prendas combinan.

De esta forma, se considerarán los modelos de los trabajos *Learning Type-Aware Embeddings for Fashion Compatibility* y *Fashion Outfit Complementary Item Retrieval* como los modelos base del trabajo.

4.3. Definición de modelos

Una vez realizado el filtro de los modelos del estado del arte que no cumplen las tareas y requisitos, se definen en detalle los modelos de recomendación que se evalúan y comparan.

Los modelos de recomendación de ropa a evaluar utilizan redes neuronales multicapas para calcular una vectorización no lineal para un ítem de dato x_i. El objetivo es aprender los parámetros θ del mapeo f_θ de tal manera que, para un par de ítem (x_i, x_j), la distancia euclidiana entre los vectores de representaciones y_i e y_j refleje su compatibilidad. El propósito es obtener un espacio de representaciones donde la distancia sea pequeña entre ítem considerados compatibles y grande entre ítem que no son compatibles.

Los modelos supervisados considerados asumen que existe una taxonomía de τ tipos, donde el tipo de un ítem se denota como un superíndice, es decir, $x_i^{(t)}$ representa el ítem i de la categoría t, donde $t = 1, \ldots, \tau$. Una tripleta se define como un conjunto de imágenes $x_i^{(u)}, x_j^{(v)}, x_k^{(v)}$ con la siguiente relación: el par (x_i, x_j) es compatible, lo que significa que dos

ítem aparecen juntos en una vestimenta, mientras que x_k es un ítem seleccionado al azar del mismo tipo que x_j pero que no ha sido visto en una vestimenta. La idea detrás del uso de esta tripleta es que el modelo minimice la distancia entre las vectorizaciones de las imágenes compatibles en el espacio de representaciones y aleje las vectorizaciones de las imágenes no compatibles. Así, la función de pérdida de tripleta se define de forma estándar como:

$$l(i, j, k) = \text{máx}\{0, d(i, j) - d(i, k) + \mu\}$$

Donde μ es un margen.

4.3.1. Type-Aware Model

Denominaremos $M(u, v)$ como el espacio de representaciones específico para los tipos de datos u y v, donde se emparejan objetos de estos tipos. En este espacio, se encuentra una proyección $P_{u\to(u,v)}$ que mapea la representación de un objeto del tipo u a $M(u, v)$. Luego, para un par de datos $(x_{(u)i}, x_{(v)j})$ que son compatibles, se requiere que la distancia entre las proyecciones sea pequeña. Esta se define como:

$$\|P_{u\to(u,v)}(f(x_{(u)i}; \theta)) - P_{v\to(u,v)}(f(x_{(v)j}; \theta))\|$$

Este enfoque general implica el aprendizaje de $2(n \times n)$ matrices por cada par de tipos, considerando una representación general de n dimensiones. Se asume que estas matrices de proyección son diagonales, es decir, $P_{u\to(u,v)} = P_{v\to(u,v)} = \text{diag}(w(u, v))$, donde $w(u, v) \in \mathbb{R}^n$ es un vector binario fijo seleccionado de antemano para cada espacio específico de tipo a par, actuando como una función de puerta que elige las dimensiones relevantes de la representación, las cuales son más responsables para determinar la compatibilidad. La medida de compatibilidad se calcula como:

$$d_{ij}^{uv} = d(x_{(u)i}, x_{(v)j}, w(u, v)) = \|f(x_{(u)i}; \theta) \odot w(u, v) - f(x_{(v)j}; \theta) \odot w(u, v)\|^2$$

donde $\odot$ denota la multiplicación componente a componente. Este cálculo se aprende mediante una pérdida de tripleta modificada, donde μ es un margen:

$$L_{comp}(x_{(u)i}, x_{(v)j}, x_{(v)k}, w(u, v); \theta) = \text{máx}\{0, d_{ij}^{uv} - d_{ik}^{uv} + \mu\}$$

$$\text{con } d_{ij}^{uv} = d(x_i^u, x_j^{(v)}, w^{(u,v)}) = ||f(x_i; \theta) \cdot w^{(u,v)} - f(x_j; \theta) \cdot w^{(u,v)}||^2$$

Como se puede apreciar en la Figura 15, el funcionamiento del modelo es el siguiente:

1. Se introduce una entrada al modelo, compuesta por una imagen de una prenda, la categoría a la que pertenece y la categoría de las prendas que se buscan.
2. Se obtiene una representación general de la prenda usando el *backbone*.

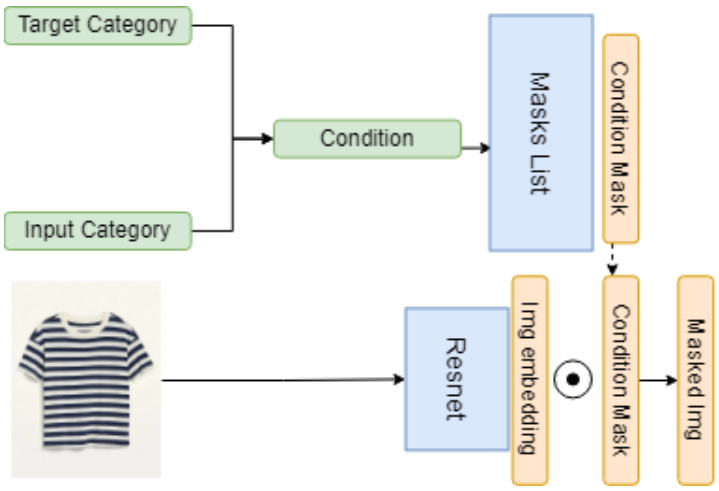

Figura 15: Arquitectura del modelo *Type-Aware*

3. Simultáneamente, se alimenta la concatenación del *one-hot-encoding* de las categorías en una matriz de máscara. Esto permite la selección de la máscara correspondiente que se aplicará a la representación de la imagen. Dicha máscara selecciona las dimensiones pertinentes de la representación para cada par de categorías.
4. Se proyecta la representación general de la imagen al subespacio específico del par de categorías.

Para regularizar la noción de compatibilidad, se utiliza la información de las descripciones de texto que acompañan a cada imagen, las cuales se alimentan como entrada a una red de vectorización de texto. La vectorización resultante de la red para la descripción $t_i^{(u)}$ de la imagen $x_i^{(u)}$ se denota como $g(t_i^{(u)}; \phi)$. Reemplazando g por f en la función l según se requiere, la pérdida para aprender similitud se escribe como:

$$L_{sim} = \lambda_1 l(x_j^{(v)}, x_k^{(v)}, x_i^{(u)}) + \lambda_2 l(t_j^{(v)}, t_k^{(v)}, t_i^{(u)})$$

El modelo también entrena una vectorización visual-semántica basándose en el trabajo de Han et al. [21], que requiere que la imagen $x_i^{(u)}$ tenga una vectorización cercana a su descripción $t_i^{(u)}$ en el espacio visual-semántico, al igual que las descripciones de las otras dos imágenes en la tripleta.

$$L_{vsei} = l(x_i^{(u)}, t_i^{(u)}, t_j^{(v)}) + l(x_i^{(u)}, t_i^{(u)}, t_k^{(v)})$$

Por último, para fomentar la dispersión en los pesos aprendidos w y contribuir a la compatibilidad de tipos de pares, se agrega una penalización en las matrices de proyección. Además, se utiliza una regularización l_2 en los vectores generales aprendidos de la imagen $f(x; \theta)$. La función de pérdida final se computa como:

$$L(X, T, P^{\cdot \to (.,.)}, \lambda, \theta, \phi) = L_{comp} + L_{sim} + \lambda_3 L_{vse} + \lambda_3 L_{l_2} + \lambda_5 L_{l_1}$$

4.3.2. Fully-Connected Type Aware Model

Corresponde a una variante del modelo descrito anteriormente, con una arquitectura y funcionamiento muy similares. Similar al modelo anterior, se introduce una entrada compuesta por una imagen de una prenda, la categoría a la que pertenece y la categoría de las prendas que se buscan. Se obtiene una representación general de la prenda usando una ResNet pre-entrenada y se alimenta la concatenación del *one-hot-encoding* de las categorías a una red neuronal. La diferencia radica en que, en este caso, la red neuronal que procesa los vectores de categorías es una capa completamente conectada para su proyección específica del tipo. Es decir, el *condition mask* es una capa *fully-connected*. Esta posee la ventaja de capturar relaciones no lineales complejas entre las características de entrada y salida.

El modelo también regulariza la noción de compatibilidad usando la información de las descripciones de texto que acompañan a las imágenes según similitud, usando las descripciones y penalizando las matrices de proyección.

4.3.3. CSA Adapt Model

Se implementa el modelo introducido en el trabajo de Son et al. [22] con algunas diferencias. En este modelo se presenta una red de atención sub-espacial que se basa en la concatenación de categorías, y se utilizan varios subespacios de estilos para capturar múltiples dimensiones de similitud. El método de atención recibe las categorías de la imagen de entrada y la categoría objetivo para ponderar la relevancia de cada subespacio en la computación de la vectorización final. En este enfoque, en lugar de utilizar la función de pérdida de vestimentas se implementa la función de pérdida de tripletas.

Como se puede apreciar en la Figura 16, el funcionamiento del modelo es el siguiente:

1. Se introduce una entrada compuesta por una imagen de una prenda, la categoría a la que pertenece y la categoría de las prendas que se buscan.
2. Se obtiene una representación general de la prenda utilizando el *backbone*. Esta representación se multiplica por un conjunto de máscaras $(m_1, ..., m_k)$ para obtener las representaciones de subespacios que encapsulan distintas características.
3. La concatenación de los dos vectores de categoría se utiliza para predecir los pesos de atención de los subespacios $(w_1, ..., w_k)$, que tienen la función de ponderar los vectores de los subespacios para el cálculo final de la representación. La red subespacial de atención está compuesta secuencialmente por una capa *fully-connected*, una función de activación *relu*, una segunda capa *fully-connected* y finalmente una función *softmax*.
4. La vectorización final se calcula como una suma ponderada de los vectores de subespa-

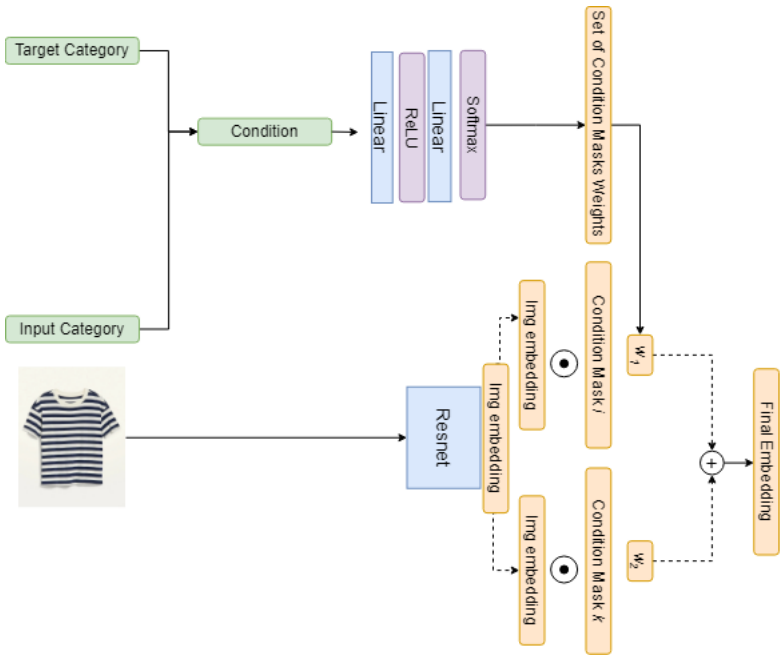

Figura 16: Arquitectura del modelo CSA Adapt

cio:

$$f = \sum_{i=1}^{k} (x \cdot m_i) \cdot w_i$$

donde k corresponde al número de subespacios, x es el vector de características, w_i son los pesos de atención y f es el vector final.

4.3.4. CSA Adapt Fully Connected Model

Se implementa un modelo similar al anterior, pero se introduce una capa *fully-connected* con un peso de tamaño (p, k) para cada vector x correspondiente a cada condición. El propósito de esta es capturar relaciones no lineales complejas entre las características de entrada y salida de cada condición, con el fin de obtener mejores representaciones.

4.3.5. CSA Net con Roberta Attention Model

Se implementa un modelo similar al CSA Adapt Model, pero en lugar de utilizar el mecanismo de atención basado en la concatenación de las representaciones *one-hot-encoding* de las categorías, se emplea un mecanismo de atención basado en la concatenación de las representaciones según un modelo de lenguaje natural.

El modelo utilizado es un modelo Roberta pre-entrenado en diversos conjuntos de datos. Este modelo mapea oraciones y párrafos a un espacio vectorial denso de 768 dimensiones y puede ser utilizado para tareas como agrupación o búsqueda semántica.

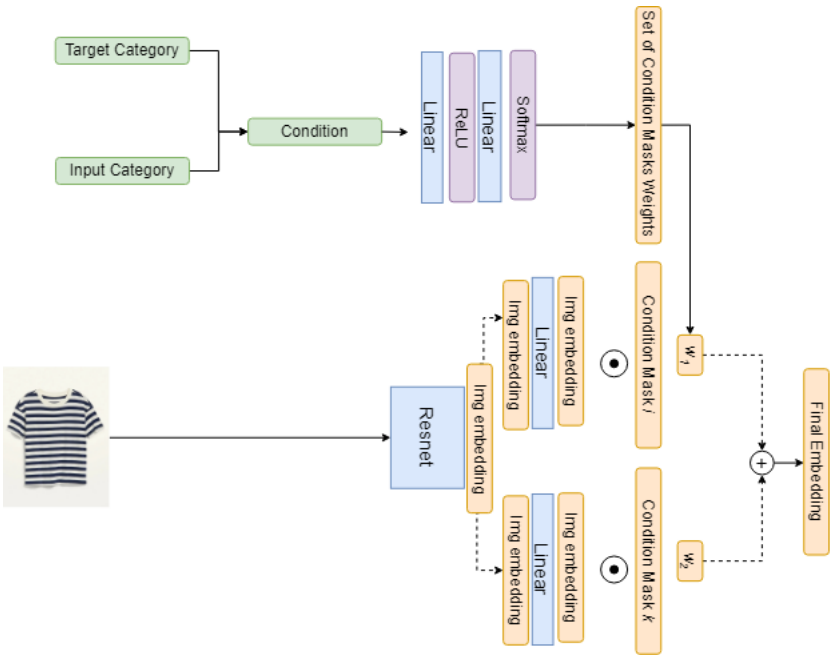

Figura 17: Arquitectura del modelo CSA Adapt Fully Connected

4.3.6. BYOL-DualNet

Como se explicó en la sección de marco teórico, el modelo original de BYOL toma una imagen como entrada y se entrena utilizando dos vistas aumentadas de la misma imagen, las cuales llamaremos v_1 y v_2. En este proceso, el modelo utiliza la red *target* para procesar la vista v_1 y la red *online* para procesar la vista v_2, generando así sus respectivas representaciones. Luego, el modelo realiza una predicción de la segunda representación basándose en la primera.

En nuestra variante BYOL-DualNet, el modelo recibe dos imágenes como entrada. Durante el entrenamiento, se trabajan con pares de imágenes compatibles, lo que elimina la necesidad de seleccionar ejemplos negativos para cada caso. Esta adaptación permite simplificar el proceso de entrenamiento y mejora la eficiencia al no requerir la búsqueda y selección de ejemplos negativos en cada iteración.

El funcionamiento del modelo es el siguiente:

1. La imagen 1 se introduce en la red en línea para calcular su representación y luego aplicar una proyección. La obtención de la representación implica el uso de un *backbone*, mientras que la proyección se logra mediante un *multi-layer perceptron* que comprende una capa lineal seguida de un *batch normalization*, funcion de activación *ReLU* y una última capa lineal con una dimensión final de 256.
2. La imagen 2 se proporciona a la red *target* para calcular su representación y posteriormente aplicar una proyección. Es importante destacar que la estructura de la red *target* es similar a la de la red en línea, con la diferencia de que sus parámetros son una versión suavizada de los parámetros de la otra red. En este caso, se trata de una media móvil exponencial de los parámetros de la red original.

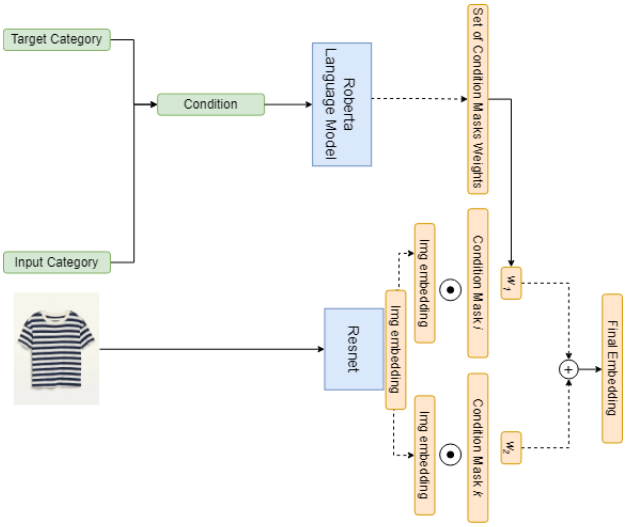

Figura 18: Arquitectura del modelo CSA-Net con Roberta Attention

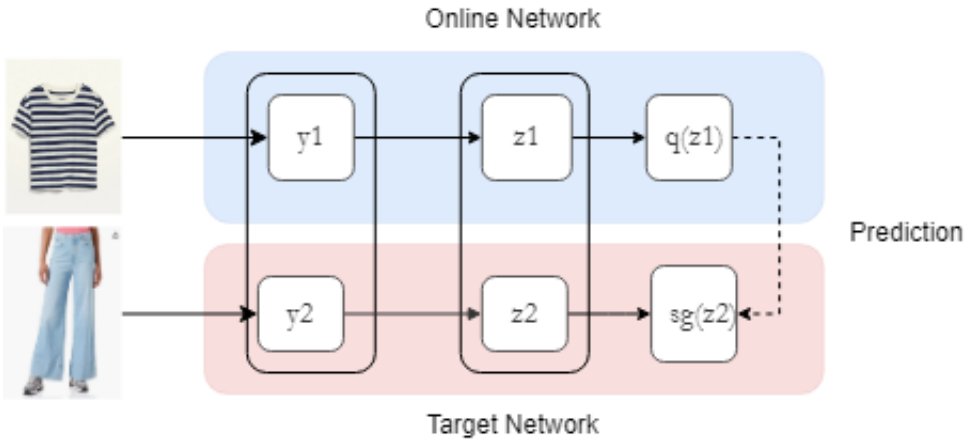

Figura 19: Arquitectura del modelo BYOL Dual-Net

3. Se calcula la predicción de la imagen 1 con respecto a la imagen 2, seguida de una normalización l2. La estructura de la predicción es similar a la de la proyección, consistiendo en un perceptrón multicapa compuesto por una capa lineal, seguida de un *batch normalization*, funcion de activación *ReLU* y una última capa lineal de dimensionalidad 256.
4. Se calcula el error cuadrático medio entre la predicción de la imagen 1 y la proyección normalizada de la imagen 2, el cual se utiliza para entrenar y evaluar el rendimiento del modelo.

Es importante destacar que no se aplican las transformaciones que se utilizan en la implementación original de BYOL debido a la diferencia del problema que estamos tratando. Para obtener las representaciones de las imágenes, se entregan a la red *online* y se capturan solo las representaciones y_1. Es importante destacar que se utiliza la configuración del modelo utilizada en el trabajo original [7].

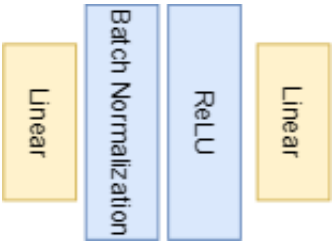

Figura 20: Arquitectura de las capas MLP de BYOL

4.3.7. Simple Siamese Dual-Net

El modelo *Simple Siamese* opera de manera similar al modelo BYOL y se adapta de la misma manera. Se implementa y configura el modelo *Simple Siamese* para que reciba dos imágenes de entrada durante el entrenamiento, las cuales formarán los pares de ejemplos positivos.

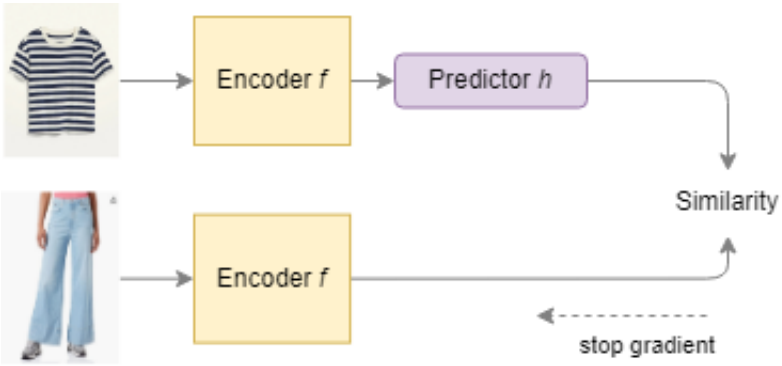

Figura 21: Arquitectura del modelo Simple Siamese Dual-Net

Como se explicó en la sección de marco teórico, el codificador f corresponde a un *backbone* con una proyección MLP, que posee una arquitectura similar a la proyección MLP del modelo BYOL en la Figura 20. Se utiliza una dimensionalidad de vectores finales de 2048 y de proyección de 512, tal como se indica en el trabajo original [20]. Por último, se mide la similitud y se minimiza la pérdida usando *Negative Cosine similarity*. Para recuperar las representaciones de las prendas con el modelo ya entrenado, se utiliza solo el codificador f.

4.4. Sistema de recuperación

Se implementa un sistema de recuperación de prendas para cada uno de los modelos anteriores. La recuperación se entiende como la tarea de encontrar ítems compatibles para completar una vestimenta. Para el conjunto a definir, se computan las representaciones para todas las imágenes y se compara la compatibilidad entre dos prendas usando distancia euclidiana; mientras más cerca sus vectorizaciones en el espacio de representaciones más compatibles son las prendas. Cabe destacar que los modelos supervisados computan vectores para distintos pares de categorías, por lo que se buscan las prendas objetivo en el subespacio

correspondiente.

Así, dada una imagen de una prenda, la categoría a la que pertenece, la categoría de la prenda que se quiere buscar y la cantidad de prendas a retornar (denotados x_i, t_i, t_j, k), el sistema de recuperación funciona de la siguiente forma:

1. Dado un modelo ya entrenado, se computan todas las vectorizaciones de las imágenes del conjunto de testeo, y se guardan junto a su identificación en la base de datos y la categoría a la que pertenecen.
2. Se filtran todas las imágenes que no sean de categorías t_j.
3. Se calcula la distancia euclidiana entre las representaciones de la imagen de la prenda y todas las imágenes filtradas.
4. Se ordenan en orden descendente y se retornan las k primeras prendas.

4.5. Conjunto de Datos y Evaluación

En esta sección se describe el conjunto de datos utilizado en los experimentos y se explica cómo se llevó a cabo la evaluación de los modelos de recomendación de moda:

4.5.1. Métricas de evaluación

- Fill-in-the blank - FITB

  En esta tarea de preguntas y respuestas, el objetivo es seleccionar la elección correcta entre cuatro candidatos, dadas las opciones incompletas de compatibilidad de ropa.

Figura 22: Ejemplo de la tarea *Fill-in-the-blank* [21]

  Para cada atuendo, se selecciona al azar un ítem y se reemplaza con un espacio en blanco. Luego, se eligen tres ítems de otras vestimentas junto con el elemento verdadero, para formar un conjunto de opciones múltiples. Se parte del supuesto de que un ítem seleccionado al azar debe ser menos compatible que aquel elegido por diseñadores profesionales. Se entenderá esta tarea bajo el acrónimo FITB.

- Positional fill-in-the-blank - Pos FITB

  En este trabajo se introduce la métrica *Positional fill-in-the-blank*, que considera la posición de la elección correcta dentro de las alternativas de respuesta en la tarea de

Fill-in-the blank. Se obtiene el puntaje de compatibilidad para cada vestimenta variando la alternativa en el espacio en blanco y se ordenan de mayor a menor. Se computa un valor entre 0 y 1 dependiendo de la posición entre la cantidad total de alternativas. Se entenderá esta tarea bajo el acrónimo Pos FITB.

- *Fashion Compatibility* - FC

  Fashion compatibility es una tarea binaria donde el modelo tiene que predecir si una vestimenta es compatible o no. Se entenderá esta tarea bajo el acrónimo FC.

Figura 23: Ejemplo del funcionamiento de *Fashion Compatibility* [21]

4.5.2. Conjunto de Datos

Se trabaja con el conjunto de datos *Polyvore Outfits*. Los datos se recompilaron a partir del sitio web *Polyvore*, una plataforma en línea donde los usuarios podían crear conjuntos de ropa y accesorios. El objetivo principal de este conjunto de datos es proporcionar una fuente de datos para tareas relacionadas con la moda, como recomendación de conjuntos, análisis de tendencias y reconocimiento de estilo.

El conjunto de datos contiene 21,889 vestimentas, donde 17,316 corresponden al entrenamiento, 1,497 para validación y 3,076 para testeo. Cada conjunto de vestimenta contiene por lo menos 4 prendas de ropa. Cada ítem de JSON tiene la siguiente información:

```
"name": Nombre de la vestimenta,
"views": Cantidad de vistas de la vestimenta,
"items": [
    Prendas en la vestimenta.
    {
        "index": Indice del ítem de moda de vestimenta en Polyvore,
        "name": Descripción de la prenda de moda,
        "price": Precio del ítem,
        "likes": Número de me gusta en el ítem,
        "image": Url de la imágen del ítem,
        "categoryid": Id de la categoría del ítem,
    },
    {
        ...
    },
    ...
],
```

```
"image": Url de la imágen de la vestimenta,
"likes": Cantidad de me gusta de la vestimenta,
"date": Fecha de carga de la vestimenta,
"set_url": Url del outfit,
"set_id": Id del outfit,
"desc": Descripción de la vestimenta.
```

El conjunto de datos también tiene archivos *category_.txt* que contiene un mapeo entre el id de una categoría y el nombre de la categoría. También contiene un archivo de preguntas usadas para evaluar la tarea de recomendación *fill-in-the-blank* que sigue el siguiente formato:

```
"question": Secuencia de articulos de moda para formar la pregunta,
"answers": Conjunto de opciones múltiples para elegir,
"blank_position": La posición en blanco que se debe completar.
```

Por último, también contiene un archivo *fashion-compatibility-prediction.txt* con 7,000 vestimentas, donde 4,000 son incompatibles y 3,000 son compatibles. En cada línea, el primer número indica la compatibilidad (1 es compatibles y 0 es incompatible) seguido por una secuencia de ítems consistiendo la vestimenta.

4.6. Experimentos

Se llevaron a cabo experimentos con los modelos definidos, manteniendo ciertos parámetros constantes y variando otros para evaluar el rendimiento y el sistema de recuperación de recomendaciones de moda. La implementación de los modelos se lleva cabo con el lenguaje de programación *Python* y la librería *Pytorch* para construir la arquitectura de los modelos de aprendizaje. Se ha empleado el repositorio de mvasil y BryanPlummer como punto de partida para el proyecto: fashion-compatibility.

Los parámetros que se mantuvieron constantes durante los experimentos son los siguientes:

- Cantidad de épocas: Los modelos se entrenaron durante un total de 10 épocas. Se registraron los resultados del modelo para cada época en los datos de validación y se seleccionó el modelo con los mejores resultados. Es importante destacar que los modelos no mejoran significativamente sus resultados para más de 10 épocas.
- Learning rate: Este hiperparámetro controla el porcentaje de cambio con el que se actualizan los pesos de la red en cada iteración del entrenamiento. Para todos los modelos posee un valor inicial constante igual a $5e^{-5}$ y decrece exponencialmente en un factor 0,015 en cada época.
- Optimizador: Se utiliza un optimizador *Adam* para ajustar los parámetros del modelo en cada iteración.

Las variables que se variaron durante los experimentos son las siguientes:

- Tamaño del mini-lote en entrenamiento: Se probaron tres tamaños diferentes para el mini-lote en entrenamiento, específicamente 96, 128 y 256. Esto permitió evaluar cómo el tamaño del lote afecta el rendimiento del modelo y la eficiencia del entrenamiento.
- Dimensionalidad del vector final: Se evaluaron dos opciones para la dimensionalidad del vector final del modelo, con tamaños de 64 y 128. Esto se hizo para analizar cómo la dimensionalidad influye en la capacidad del modelo para capturar características relevantes de los atuendos. Esta variablilidad solo se mide en los modelos que utilizan ejemplos negativos. Para los modelos Byol Dual-Net y *Simple Siamese Dual-Net* se utiliza la configuración que entrega los mejores resultados en el trabajo original.

Es esencial resaltar que los tiempos de entrenamiento de los modelos no se han comparado, ya el entrenamiento para 10 épocas requiere muy poco tiempo. Para comparar los resultados del sistema de recuperación, se elige el par (tamaño del batch, dimensionalidad) óptimo para cada modelo y se comparan los resultados para una imágen y categoría objetivo iguales.

4.7. Construcción de ejemplos positivos y negativos

Como se ha mencionado, ciertos modelos requieren tanto ejemplos positivos como negativos para su entrenamiento, mientras que otros solo necesitan ejemplos negativos. En el caso del conjunto de datos de *Polyvore Outfits*, que proporciona información sobre conjuntos completos de ropa junto con las prendas individuales que los componen, los pares de ejemplos positivos se construyen considerando todas las combinaciones posibles de prendas en un atuendo. Por ejemplo, para una vestimenta compuesta por tres prendas de las categorías c_i, c_j y c_k (que podrían representar *prendas superiores*, *prendas inferiores* y *zapatos* respectivamente), se pueden generar ejemplos positivos de los pares de categorías (c_i, c_j), (c_j, c_k) y (c_i, c_k).

Las prendas negativas de categoría c_j para una categoría c_i se construyen de forma aleatoria: a partir de todas las prendas del conjunto de datos de la categoría c_j, se elige un ejemplo al azar y se emplea como ejemplo negativo. Esto se basa en la suposición de que las prendas entre diferentes vestimentas tienden a combinar menos que las prendas dentro de una misma vestimenta.

4.8. Construcción de interfaz para visualización de los modelos de recomendación

Se implementa una interfaz web para visualizar los resultados del sistema de recuperación de los modelos más relevantes, para que así sean accesibles para futuros trabajos.

La construcción se lleva a cabo utilizando el lenguaje de programación *Python* y usando la tecnología *Django* como *framework* de desarrollo web. La aplicación consiste en dos vistas o pestañas. La primera es la extensión "*/home*" y, como se puede visualizar en la Figura 24,

describe los modelos de recomendaciones disponibles y el funcionamiento de los modelos de recomendación. La entrada de los modelos de recomendación consiste en:

Figura 24: Interfaz de la pestaña "home/"

1. Imagen de la prenda en formato *.jpg* o *.png*.
2. Categoría de la prenda de imagen de entrada que se selecciona de un menú desplegable.
3. Categoría de la prenda objetivo, es decir, que se quiere recuperar.
4. Cantidad de prendas a recuperar y retornar por el sistema.

La segunda pestaña "*/models*" permite, como se puede observar en la Figura 25, seleccionar el modelo a revisar aparte de las entradas mencionadas anteriormente para revisar los resultados del modelo de recuperación. Entre el listado de modelos a seleccionar se encuentran:

- CSA Adapt Model
- Simple Siamese Dual-Net

Se elige el modelo con la selección de parámetros e hiperparámetros, y entrenado en la cantidad de épocas que optimiza el rendimiento del modelo en las métricas de evaluación.

El sistema de recuperación de la interfaz se implementa de la siguiente forma:

1. Se entrenan los modelos de recomendación con la combinación de paramétros e hiperparametros que obtiene los mejores resultados en las métricas de evaluación.
2. Se aplican las transformaciones pertinentes al conjunto de datos de evaluación, y se obtienen las representaciones para cada modelo.
3. Se guardan los modelos junto con las representaciones de todas las imágenes, y la combinación de parámetros e hiper-parámetros para la recuperación en la aplicación.

Figura 25: Interfaz de la pestaña "models/"

Continuando en la aplicación de *Django*, se guardan los modelos un directorio accesible por la aplicación, y dados el nombre de un modelo m, la imagen de una prenda im_i, categoría de la prenda c_i, categoría objetivo c_j y cantidad de prendas a recuperar n, se siguen los siguientes pasos:

1. Se cargan los hiper-parámetros y parámetros para el modelo especificado.
2. Se inicializa un modelo con los hiper-parámetros y parámetros óptimos, y se carga el modelo recuperándolo del directorio de modelos guardados.
3. Se cargan todas las representaciones de todas las imágenes junto a su ubicación en el directorio de imágenes de *Polyvore Outfits*.
4. Se filtran las representaciones que no pertenezcan al sub-espacio de prendas (c_i, c_j), y todas las representaciones de prendas que no sean de categoría c_j.
5. Usando distancia euclideana se calcula la distancia entre la representación de la imagen de entrada y todas las representaciones filtradas, y se ordenan de menor a mayor.
6. Se eligen las n imágenes con menor distancia euclidiana y se retornan para visualizar en la aplicación. Para esto se utiliza la librería *matplotlib*.

Capítulo 5

Resultados Experimentales y Análisis

5.1. Resultados de los experimentos

Los resultados obtenidos de los experimentos se presentan en la siguiente tabla. El mejor modelo será el que obtenga los mejores resultados en la siguiente métrica:

$$f_{MM} = \text{FC} * 100 + \text{FITB} + \text{Pos FITB}$$

1. Modelo Type-aware

Tabla 1: Resultados de los experimentos del modelo Type-aware

Tamaño del Mini-lote	Dimensionalidad del Vector	FC	FITB	Pos FITB
96	64	0.89	59.8	63.0
96	128	0.88	58.1	62.3
128	64	0.89	58.8	62.9
128	128	0.86	57.2	62.5
256	64	0.85	55.6	62.5
256	128	0.87	57.2	62.3

Se considera así el mejor modelo al que posee los parámetros:
(Tamaño del mini-lote, Dimensionalidad del vector) = $(96, 64)$.

2. Modelo Type-aware *Fully Connected*

Tabla 2: Resultados de los experimentos del modelo Type-aware *Fully Connected*

Tamaño del Mini-lote	Dimensionalidad del Vector	FC	FITB	Pos FITB
96	64	0.87	56.5	62.9
96	128	0.88	57.8	62.5
128	64	0.86	56.8	62.4
128	128	0.86	57.1	62.1
256	64	0.86	56.0	62.2
256	128	0.86	56.2	62.1

Se considera así el mejor modelo al que posee los parámetros:
(Tamaño del mini-lote, Dimensionalidad del vector) = $(96, 128)$.

3. Conditional Siamese Network

Tabla 3: Resultados de los experimentos del modelo Conditional Siamese Network

Tamaño del Mini-lote	Dimensionalidad del Vector	FC	FITB	Pos FITB
96	64	0.85	55.9	62.1
96	128	0.86	56.5	62.3
128	64	0.85	54.8	62.1
128	128	0.86	55.8	62.1
256	64	0.85	54.3	61.9
256	128	0.86	55.4	62.2

Se considera así el mejor modelo al que posee los parámetros:
(Tamaño del mini-lote, Dimensionalidad del vector) = $(96, 128)$.

4. Conditional Siamese *Fully Connected* Network

Tabla 4: Resultados de los experimentos del modelo Conditional Siamese *Fully Connected* Network

Tamaño del Mini-lote	Dimensionalidad del Vector	FC	FITB	Pos FITB
96	64	0.86	54.3	62.2
96	128	0.85	53.6	61.4
128	64	0.84	53.5	61.9
128	128	0.85	53.5	61.9
256	64	0.84	53.4	61.6
256	128	0.85	55.3	61.9

Se considera así el mejor modelo al que posee los parámetros:
(Tamaño del mini-lote, Dimensionalidad del vector) = $(96, 64)$.

5. Conditional Siamese Roberta Network

Tabla 5: Resultados de los experimentos del modelo Conditional Siamese Roberta Network

Tamaño del Mini-lote	Dimensionalidad del Vector	FC	FITB	Pos FITB
96	64	0.86	55.9	62.4
96	128	0.86	55.5	62.0
128	64	0.85	55.4	62.0
128	128	0.86	54.3	61.7
256	64	0.86	55.2	62.2
256	128	0.86	54.0	62.0

Se considera así el mejor modelo al que posee los parámetros:
(Tamaño del mini-lote, Dimensionalidad del vector) = $(96, 64)$.

6. BYOL Dual-Network

El mejor modelo corresponde al que posee los parámetros:
(Tamaño del mini-lote) = (512).

Tabla 6: Resultados de los experimentos del modelo BYOL Dual-Network

Tamaño del Mini-lote	FC	FITB	Pos FITB
256	0.58	33.4	53.7
512	0.58	33.7	53.8

7. Simple Siamese Dual-Network

Tabla 7: Resultados de los experimentos del modelo Conditional Siamese Roberta Network

Tamaño del Mini-lote	FC	FITB	Pos FITB
256	0.66	40.5	56.4
512	0.65	36.4	55.4

El mejor modelo corresponde al que posee los parámetros:
(Tamaño del mini-lote) = (256).

5.2. Sistema de recuperación

Se examinan los resultados del sistema de recuperación para ocho ejemplos, en los cuales se presentan los resultados correspondientes a una categoría objetivo para cada uno de ellos.

(a) Top (b) Pulsera (c) Bottom (d) Bags (e) Shoes

Figura 26: Conjunto de imágenes de prendas etiquetadas (a) a (e).

1. En la Figura 26a se presenta una prenda de categoría *tops* y se recuperan prendas de categoría *shoes*.
2. En la Figura 26b se presenta una prenda de categoría *jewellery* y se recuperan prendas de categoría *tops*.
3. En la Figura 26c se presenta una prenda de categoría *bottoms* y se recuperan prendas de categoría *accessories*.
4. En la Figura 26d se presenta una prenda de categoría *bags* y se recuperan prendas de categoría *hats*.
5. En la Figura 26e se presenta una prenda de categoría *shoes* y se recuperan prendas de categoría *shoes*.

5.2.1. Modelo Type-Aware

Figura 27: Resultados del modelo Type-aware

5.2.2. Modelo Type-Aware *Fully-Connected*

Figura 28: Resultados del modelo Type-aware *Fully-Connected*

5.2.3. Modelo CSA-Adapt

Figura 29: Resultados del modelo CSA-Adapat

5.2.4. Modelo CSA-Adapt *Fully Connected*

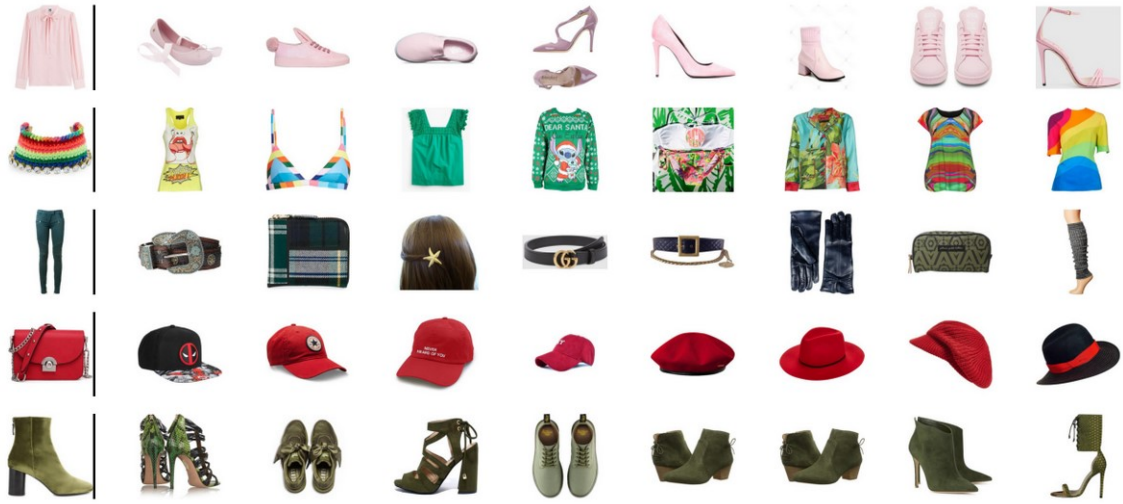

Figura 30: Resultados del modelo CSA-Adapat *Fully-Connected*

5.2.5. Modelo CSA-Adapt con Roberta Language Model

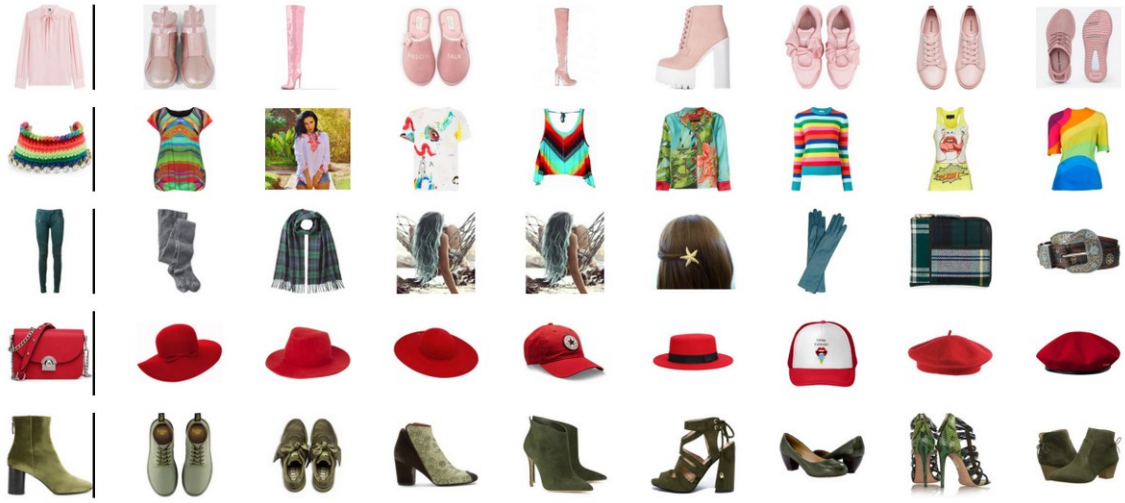

Figura 31: Resultados del modelo CSA-Adapt con Roberta Language Model

5.2.6. Modelo BYOL Dual-Net

Figura 32: Resultados del modelo BYOL Dual-Net

5.2.7. Modelo SimSiamese Dual-Net

Figura 33: Resultados del modelo BYOL Dual-Net

A continuación se muestran algunos resultados de los mejores modelos CSA-Adapt y SimSiamese Dual-Net. Se eligen estos modelos puesto que consiguen los mejores resultados en las métricas de evaluación.

5.3. Más ejemplos de CSA-Adapt

Figura 34: Ejemplos del sistema de recuperación de CSA-Adapt 1

Figura 35: Ejemplos del sistema de recuperación de CSA-Adapt 2

Figura 36: Ejemplos del sistema de recuperación de CSA-Adapt 3

Figura 37: Ejemplos del sistema de recuperación de CSA-Adapt 4

Figura 38: Ejemplos del sistema de recuperación de CSA-Adapt 5

5.4. Más ejemplos de SimSiam Dual-Net

Figura 39: Ejemplos del sistema de recuperación de Simple Siamese 1

Figura 40: Ejemplos del sistema de recuperación de Simple Siamese 2

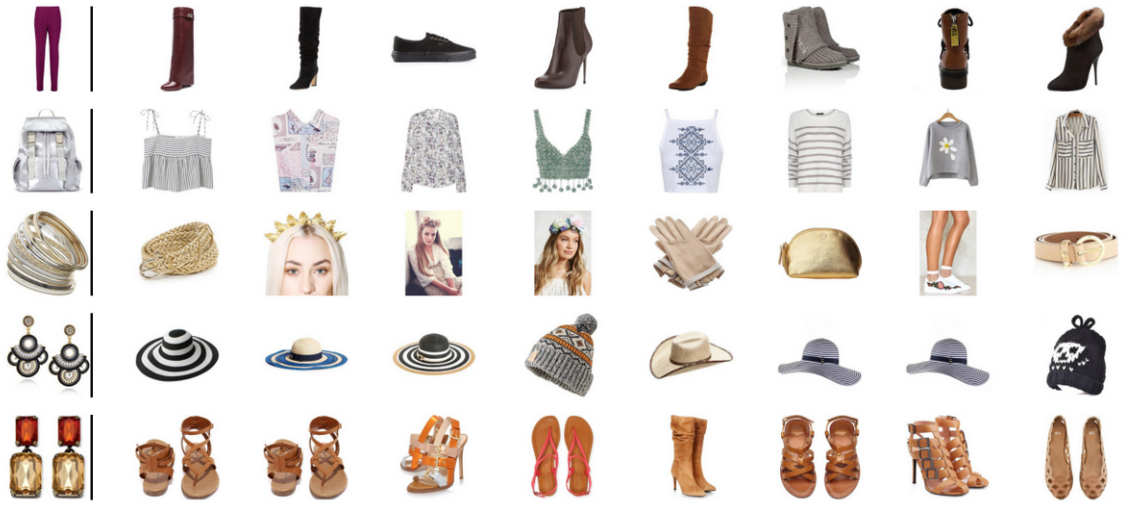

Figura 41: Ejemplos del sistema de recuperación de Simple Siamese 3

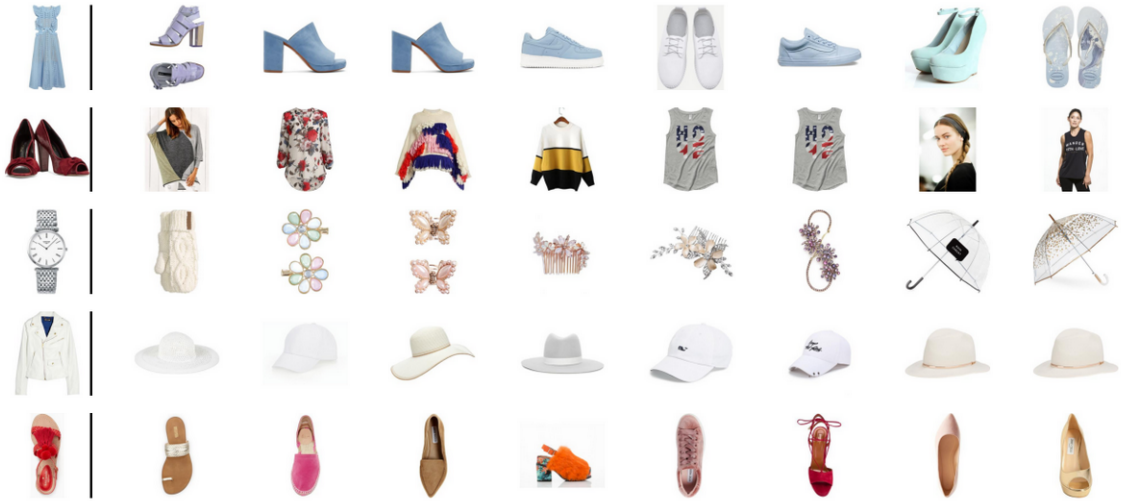

Figura 42: Ejemplos del sistema de recuperación de Simple Siamese 4

Figura 43: Ejemplos del sistema de recuperación de Simple Siamese 5

5.5. Análisis

5.5.1. Modelos Entrenados con Ejemplos Positivos y Negativos

En primer lugar, es relevante destacar que los modelos entrenados con ejemplos positivos y negativos exhiben un rendimiento superior cuando se emplea un tamaño de lote reducido. Este fenómeno puede atribuirse al hecho de que los tamaños de lotes más pequeños son ruidosos, por lo que ofrecen un efecto de regularización y menor error de generalización. En contraste, la dimensionalidad del vector no tiene un impacto significativo en el desempeño de los modelos.

Tabla 8: Resultados de los modelos bajo con la mejor configuración.

Modelo	FC	FITB	Pos FITB
Type-aware	0.89	59.8	63.0
Type-aware *Fully Connected*	0.88	57.8	62.5
CSA-Adapt	0.86	56.5	62.3
CSA-Adapt Fully Connected	0.86	54.3	62.2
CSA-Adapt con Roberta Language Model	0.86	55.9	62.4
BYOL Dual-Net	0.58	33.7	53.8
SimSiamese Dual-Net	0.66	40.5	56.4

Es interesante destacar que la implementación de la red Siamesa Condicional (*Conditional Siamese Network*) arroja resultados similares a los reportados en el estudio original. En particular, es crucial subrayar que esta implementación logra un rendimiento comparable al del modelo *Type-aware*, incluso sin la inclusión de descripciones de prendas o la aplicación de regularización de vectores mediante un espacio visual-semántico. Los detalles específicos de los modelos originales se presentan en la tabla 9.

Tabla 9: Resultados de los modelos originales según lo informado por Son et al. [22].

Método	*backbone*	FITB	FC
Siamese-Net	ResNet18	52.9	0.81
Type-aware	ResNet18 + Text	57.83	0.87
SCE-Net	ResNet18 + Text	59.07	0.88
CSA-Net + Outfit Ranking Loss	ResNet18	63.73	0.91
CSA-Net + Triplet Loss	ResNet18	60.91	0.9

La tabla presenta los resultados del modelo CSA-Net al emplear la función de pérdida *Outfit Ranking Loss* en comparación con la *Triplet Loss*. Utilizando la primera, el modelo logra mejores resultados, lo que indica que es capaz de generar representaciones más efectivas de las prendas.

En cuanto al sistema de recuperación, es relevante destacar que los modelos demuestran la capacidad de recuperar prendas similares al recibir los mismos *queries*. Es crucial notar

que las prendas recuperadas comparten similitudes en términos de patrones de colores con la imagen de entrada. Esto sugiere que el modelo otorga una considerable importancia al color de las prendas. Esta tendencia podría atribuirse al conjunto de datos de *Polyvore Outfits*, en el que las vestimentas tienden a estar compuestas por prendas que comparten un esquema de colores, como también podría relacionarse con la arquitectura inherente de los modelos.

5.5.2. Modelos Entrenados con Ejemplos Positivos

En contraste, el modelo BYOL Dual-Net no presenta resultados significativos en las métricas evaluadas. Los detalles de un modelo base BYOL Dual-Net no entrenado se exponen en la Tabla 10. En este sentido, es relevante resaltar que el modelo no obtiene resultados comparables al estado del arte bajo las condiciones establecidas. Antes de este estudio, no se habían presentado implementaciones del modelo BYOL en un formato bimodal con este enfoque. Esto se refiere a situaciones en las cuales los conjuntos de datos no sean una transformación uno del otro o carezcan de una relación visual claramente definida entre sí.

Tabla 10: Resultados del modelo base de BYOL Dual-Net

FC	FITB	FITB var
0.55	30.4	32.4

Como se detalla en la sección de Marco Teórico, BYOL se entrena utilizando vistas aumentadas de una imagen y aprende a distinguir entre estas vistas. Sin embargo, su aplicación en un contexto como el que se presenta en este trabajo no había sido explorada previamente. En este caso, el modelo es entrenado con pares de imágenes que no parecen tener una conexión visual aparente entre ellas, como lo son pares de prendas de distintas categorías.

Por otro lado, como se puede apreciar en la Tabla 8, el modelo SimSiam Dual-Net obtiene resultados más alentadores en las métricas de evaluación. Aun así, no logra resultados comparables con el estado del arte. Esta situación puede atribuirse a que la arquitectura de *Simple Siamese* no está configurada para resolver el problema de *fashion compatibility*.

Es importante destacar que la arquitectura y funcionamiento del modelo es comparable al modelo propuesto por Veit et al. [19], en el que se utilizan redes siamesas para aprender un espacio de representación unidimensional en el que se mide la compatibilidad. En este trabajo, se entrena el modelo con ejemplos positivos y ejemplos negativos seleccionados de manera aleatoria. Esta comparación pone de relieve la importancia de los ejemplos negativos en el entrenamiento de los modelos, aunque la estrategia de elección de dichos ejemplos sea aleatoria.

En relación al sistema de recuperación, se observa que los ejemplos recuperados por estos sistemas tienden a enfocarse en el color de las prendas. Sin embargo, proporcionan resultados más variados en términos de color en comparación con los modelos entrenados con ejemplos positivos y negativos.

Capítulo 6

Conclusión

En esta investigación, se cumplen los objetivos que fueron propuestos. Se llevó a cabo una evaluación exhaustiva de los modelos de recomendación basados en imágenes. Se adaptaron los modelos más importantes del estado del arte para que funcionaran con una función de pérdida basada en *Triplet Loss*, y se desarrollaron modelos que fueran entrenados con pares de imágenes de prendas compatibles. Estos modelos se evaluaron bajo las métricas FITB (*Fill-In-The-Blank*) y *fashion compatibility*, y se compararon los resultados. Además, se implementó un sistema de recuperación de prendas para analizar la efectividad de las recomendaciones. Por último, se diseñó una aplicación para visualizar y analizar los resultados de manera intuitiva.

En primer lugar, es importante destacar que los experimentos evidencian que los ejemplos negativos proveen información relevante para el entrenamiento de modelos de compatibilidad de moda. Esto habilita el aprendizaje de sistemas capaces de generar recomendaciones más precisas, incluso cuando los ejemplos negativos puedan ser recuperados mediante una estrategia subóptima. Estos ejemplos desempeñan un papel esencial en el proceso de aprendizaje y permiten que el modelo adquiera conocimiento acerca de las características fundamentales entre prendas compatibles e incompatibles.

Por lo tanto, en el entrenamiento de un modelo de recomendación de ropa, es crucial crear ejemplos positivos y negativos, y tener en cuenta una estructura que sea coherente con dichos ejemplos. Es destacable notar que este es el primer trabajo que propone medir el impacto de los ejemplos negativos en el rendimiento de los modelos de recomendación de ropa en comparación con su no uso.

En lo que respecta a los modelos supervisados, se observa que se obtienen mejores resultados cuando se utilizan tamaños de lotes más pequeños. Esto podría explicarse por la forma en que se actualizan los pesos durante el entrenamiento. Un tamaño de lote más reducido permite actualizaciones de pesos más frecuentes, lo que a su vez facilita que el modelo ajuste sus parámetros de manera más ágil a las particularidades de cada ejemplo y a las relaciones presentes en el conjunto de datos. En contraste, al utilizar lotes más grandes, el modelo podría aprender a depender excesivamente de patrones específicos en el conjunto de datos, lo que podría conducir al sobreajuste. Con respecto a la dimensionalidad de los vectores, en cambio, se observa que su influencia en el rendimiento de los modelos no parece ser significativa para una dimensionalidad mayor a 64.

Por último, al analizar los sistemas de recuperación, se observa consistencia en los resultados proporcionados por todos los modelos supervisados. Las imágenes recuperadas como recomendación generalmente muestran similitudes en términos de color en comparación con la imagen de entrada, lo que podría atribuirse tanto a la arquitectura de los modelos como

al conjunto de datos con el que fueron entrenados.

Capítulo 7

Trabajo Futuro

Como perspectivas de investigación futura, se sugieren diversas direcciones para enriquecer y avanzar en el ámbito de la modelación de compatibilidad en la moda. En primer lugar, se propone explorar transformaciones de datos que tengan el potencial de mejorar las métricas de los modelos, particularmente en lo que respecta al sistema de recuperación. La aplicación de transformaciones a los colores de las imágenes podría ser valiosa para mitigar esta problemática de sobreajuste cromático.

También, es relevante considerar la función de pérdida empleada. Aunque en este estudio se emplea una función de pérdida basada en tripletas debido al enfoque en la compatibilidad de prendas individuales, vale la pena explorar la adopción de una función de pérdida enfocada en conjuntos completos de vestimenta. Este enfoque, conocido como *outfit loss*, considera todas las prendas en un conjunto para evaluar su compatibilidad global. Dicho enfoque ha demostrado ofrecer resultados mejorados en trabajos anteriores, como lo ilustra la investigación realizada por Son et al. [22].

Por otra parte, se sugiere considerar la aplicación de modelos de recomendación basados en grafos. Esta aproximación ofrece una ventaja sustancial al abordar la problemática de la compatibilidad en la moda: su capacidad para modelar y capturar relaciones complejas y no lineales entre elementos de moda. Los métodos basados en grafos permiten representar las interacciones y conexiones entre diversas prendas de vestir de una manera más compleja. En particular, estos enfoques permiten aprovechar las relaciones contextuales entre elementos de moda. Por ejemplo, se pueden considerar elementos que han sido adquiridos juntos por usuarios o que han sido visualizados en conjunto. La incorporación de diferentes tipos de nodos y relaciones en un grafo permite modelar estas conexiones de manera efectiva, mejorando así la representación de los elementos y la capacidad de predicción de su compatibilidad.

Aunque la adopción de modelos de recomendación basados en grafos introduce una variación en la formulación del problema al requerir una comprensión de relaciones más intrincadas entre prendas en lugar de enfocarse únicamente en pares de imágenes compatibles, explorar estas técnicas puede tener un impacto significativo y justifica una investigación en profundidad.

Por último, es importante señalar que se debe evaluar el trabajo en distintos conjuntos de datos y considerando distintas métricas de evaluación para así tener un trabajo más robusto.

Bibliografía

[1] Veit A. y Kovacs Balazs. "Learning Visual Clothing Style with Heterogeneous Dyadic Co-occurrences". En: (2015). URL: https://doi.org/10.48550/arXiv.1509.07473.

[2] Popli Additya y Kumar Vijay. "Learning Fashion Compatibility from In-the-wild Images". En: (2022). URL: https://doi.org/10.48550/arXiv.2206.05982.

[3] Amazon Web Services. *¿Qué es una Red Neuronal?* Definición y estructura de las redes neuronales. URL: https://aws.amazon.com/es/what-is/neural-network/.

[4] Kim Donghyun y Saito Kuniaki. "Self-supervised Visual Attribute Learning for Fashion Compatibility". En: (2020). URL: https://doi.org/10.48550/arXiv.2008.00348.

[5] Engati. *Vanishing Gradient Problem*. Descripción del problema del desvanecimiento del gradiente. URL: https://www.engati.com/glossary/vanishing-gradient-problem.

[6] Gombru. *Understanding Ranking Loss: Contrastive, Triplet, and Beyond*. Definición y tipos de Ranking Loss. URL: https://gombru.github.io/2019/04/03/ranking_loss/.

[7] Jean-Bastien Grill y Florian Strub. "Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning". En: (2020). URL: https://doi.org/10.48550/arXiv.2006.07733.

[8] Kaiming He y Xiangyu Zhang. "Deep Residual Learning for Image Recognition". En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015). URL: https://doi.org/10.48550/arXiv.1512.03385.

[9] Sepp Hochreiter y Jürgen Schmidhuber. "Long Short-Term Memory". En: *Neural Computation* (1997).

[10] Vasileva M. I. y Plummer B. A. "Learning Type-Aware Embeddings for Fashion Compatibility". En: (2018). URL: https://doi.org/10.48550/ARXIV.1803.09196.

[11] IBM. *Convolutional Neural Networks*. Definición, estructura y usos de las redes neuronales convolucionales. URL: https://www.ibm.com/es-es/topics/convolutional-neural-networks.

[12] IBM. *Deep Learning con IBM Cloud*. Definición de aprendizaje profundo. URL: https://www.ibm.com/mx-es/cloud/deep-learning.

[13] IBM. *Redes Neuronales Recurrentes*. Definición de las redes neuronales recurrentes. URL: https://www.ibm.com/es-es/topics/recurrent-neural-networks.

[14] Microsoft Azure. *What is Artificial Intelligence?* Definición de Inteligencia Artificial. URL: https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-artificial-intelligence/.

[15] Tan Reuben y Vasileva Mariya. "Learning Similarity Conditions Without Explicit Supervision". En: (2019). URL: https://doi.org/10.48550/arXiv.1908.08589.

[16] Stuart J. Russell y Peter Norvig. *Inteligencia Artificial: Un Enfoque Moderno*. México: Pearson Educación, 1995. ISBN: 978-607-32-5050-3.

[17] Saturn Cloud. *What Does "Backbone"Mean in a Neural Network?* Importancia del backbone en una red neuronal. URL: https://saturncloud.io/blog/what-does-backbone-mean-in-a-neural-network/.

[18] Towards Data Science. *A Friendly Introduction to Siamese Networks*. Descripción y aplicación de las redes siamesas. URL: https://towardsdatascience.com/a-friendly-introduction-to-siamese-networks-85ab17522942.

[19] Andreas Veit y Balazs Kovacs. "Learning Visual Clothing Style with Heterogeneous Dyadic Co-occurrences". En: (2015). URL: https://doi.org/10.48550/arXiv.1509.07473.

[20] Chen Xinlei y He Kaiming. "Exploring Simple Siamese Representation Learning". En: (2020). URL: https://doi.org/10.48550/arXiv.2011.10566.

[21] Han Xintong y Wu Zuxuan. "Learning Fashion Compatibility with Bidirectional LSTMs". En: (2017). URL: https://doi.org/10.1145/3123266.3123394.

[22] Lin Yen-Liang y Tran Son. "Fashion Outfit Complementary Item Retrieval". En: (2019). URL: https://doi.org/10.48550/arXiv.1912.08967.