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

EVALUACIÓN DE MÉTODOS AUTO-SUPERVISADOS Y SEMI-SUPERVISADOS PARA LA EXTRACCIÓN DE CARACTERÍSTICAS VISUALES EN EL CONTEXTO DE RECUPERACIÓN DE IMÁGENES BASADA EN DIBUJOS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

JAVIER MORALES RODRÍGUEZ

PROFESOR GUÍA:
JOSÉ SAAVEDRA RONDO

PROFESOR CO-GUÍA:
NILS MURRUGARRA LLERENA

MIEMBROS DE LA COMISIÓN:
ANDRÉS ABELIUK KIMELMAN
JOSÉ URZÚA REINOSO

SANTIAGO DE CHILE
2021

Resumen Ejecutivo

La recuperación de imágenes basada en dibujos es un problema del área de visión por computadora en donde se utilizan dibujos para realizar consultas y recuperar las fotos que más se parezcan al dibujo realizado. Debido a los avances tecnológicos de la última década, este problema ha comenzado a tener relevancia en el *eCommerce*, en donde se utilizan herramientas que permiten que el usuario dibuje lo que desea comprar. En este contexto, este trabajo busca evaluar métodos auto-supervisados y semi-supervisados para la extracción de características visuales, pudiendo utilizar fotos y dibujos sin etiquetas, para entrenar modelos que resuelvan el problema descrito.

Se implementan y estudian varios modelos de redes convolucionales enfocados a extraer características de dibujos, teniendo dos conjuntos: los modelos basados en *Variational Auto-encoders* (VAE) y los modelos basados en *Bootstrap Your Own Latent* (BYOL). El primer grupo contempla modelos generativos, que son capaces de codificar una imagen en un vector de baja dimensionalidad y luego reconstruirla. Se estudian dos variedades, un VAE simple auto-supervisado que no utiliza etiquetas, y dos versiones de modelos VAE semi-supervisados capaces de ser entrenados con datos con y sin etiquetas. El primero no logra resultados que sean capaces de competir con los modelos supervisados, logrando solo un **mAP@5** de **0,310** en comparación al **0,528** obtenido por un modelo supervisado, al evaluar en un conjunto con clases distintas a las del entrenamiento. Los modelos semi-supervisados M2 y VAE semi-supervisado, logran resultados competitivos solo al evaluar con las mismas clases utilizadas en el entrenamiento, logrando un **mAP@5** de **0,648** y **0,624** respectivamente, en comparación al **0,585** obtenido por la contraparte supervisada.

En cuanto a los modelos basados en BYOL, se utiliza un BYOL simple para extraer características de dibujos, este modelo logra resultados competitivos contra modelos supervisados, logrando incluso generalizar de mejor manera hacia otras clases con un **mAP@5** de **0,590** en comparación al **0,528** obtenido por un método supervisado. Debido a esto, se diseña un modelo inspirado en BYOL para trabajar tanto con dibujos como imágenes, pudiendo enfrentar el problema de recuperación de imágenes basada en dibujos. Este modelo logra resultados competitivos con los modelos de redes siamesas que se utilizan en la actualidad, sin utilizar etiquetas ni funciones de pérdida de clasificación durante el entrenamiento, teniendo un **mAP** de **0,178** al evaluar en un conjunto de datos de *eCommerce*, en comparación al **0,145** obtenido por una red siamesa supervisada. Finalmente, se propone una extensión de método de BYOL para fotos y dibujos, en donde solo se necesitan fotos y los dibujos son generados durante el entrenamiento, por lo que sería un modelo completamente auto-supervisado.

Tabla de contenido

Índice de ilustraciones

Índice de tablas

1. Introducción

1.1. Motivación

En la última década el interés por la inteligencia artificial ha crecido de forma explosiva. Debido a los grandes avances en poder computacional y a la alta disponibilidad de datos ha sido posible aplicar técnicas de aprendizaje profundo en múltiples disciplinas, resolviendo problemas que antes parecían inalcanzables. Un área que se ha visto muy beneficiada es la visión por computadora, en donde las redes neuronales han demostrado tener un excelente desempeño, y han pasado a ser una de las herramientas más utilizadas. Hay varios sectores que se han beneficiado por este tipo de modelos, un ejemplo es el *e-commerce*, en donde se utilizan estas técnicas para mejorar sus servicios.

Un tema muy relevante en el *e-commerce* es que los usuarios sean capaces de encontrar los productos que desean comprar, poder maximizar los mecanismos con que se logra esto resulta en mayores ventas y una mejor experiencia para el consumidor. Una posible solución es incorporar técnicas de inteligencia artificial en los buscadores de productos, permitiendo incluir más información en la búsqueda que un simple texto. Una aplicación real de esto es que algunas tiendas han agregado la funcionalidad de buscar productos a través de una foto. Un nivel aún más abstracto del concepto anterior es poder dibujar lo que uno desea, el usuario no necesita haber tenido la oportunidad de conseguir una foto sino solo debe poder imaginarlo, esta es la idea detrás de la recuperación de imágenes basada en dibujos.

Para poder entrenar modelos que resuelvan el problema de recuperación de imágenes basada en dibujos, se necesita una gran cantidad de pares de dibujos e imágenes que contengan el mismo producto, además de una clasificación de qué es lo que contiene cada par. La cantidad de conjuntos de datos con estas características es bastante escasa, y la situación se complica aún más si se quiere incorporar al modelo productos que no son comunes. Por otro lado, tratar de crear un conjunto de datos que se ajuste a estas necesidades es bastante costoso pues requiere que se realicen muchos dibujos a mano. Debido a esto es interesante explorar el mundo de los modelos auto-supervisados y semi-supervisados que requieren menos información durante el entrenamiento.

Esta memoria consiste en evaluar métodos para la extracción de características de dibujos de forma auto-supervisada y semi-supervisada, estudiando la viabilidad de utilizar dibujos sin etiquetas para entrenar modelos. Se implementan arquitecturas de redes neuronales y se entrenan con dibujos hechos por personas, de forma que el modelo produzca un vector de características que describa correctamente a cada bosquejo. Para validar los resultados se comparan los vectores obtenidos con los producidos con modelos ResNet, un modelo supervisado ampliamente utilizado. Finalmente, se estudia la posibilidad de extender los modelos para incorporar imágenes y enfrentar el problema de recuperación de imágenes basada en dibujos, comparando los resultados con modelos de redes siamesas, uno de los modelos que componen el estado del arte en este problema.

1.2. Definición y relevancia del problema

La recuperación de imágenes basada en dibujos consiste en proporcionar un dibujo para realizar una consulta sobre un catálogo de imágenes para recuperar las que más se parezcan al dibujo. Impresee [1] es una empresa chilena que utiliza este concepto para construir un buscador inteligente y actualmente ofrece este servicio a compañías del retail de varios países. Este buscador permite realizar un simple dibujo sobre el producto deseado y luego encuentra los elementos del catálogo de la tienda que se parezcan a este, permitiendo que los usuarios expresen de una manera simple e intuitiva lo que desean comprar.

Actualmente, ya existen formas de abordar este problema, por ejemplo Aníbal Fuentes, en su trabajo de memoria de título [1], ha utilizado modelos de redes siamesas convolucionales para enfrentar el problema considerando dibujos con y sin color. Sin embargo, una dificultad que se menciona en este trabajo son las limitaciones debido que se presentan debido a la poca cantidad de datos disponibles con las características necesarias para entrenar los modelos, y se comenta de la posibilidad de que los resultados obtenidos fueran mejores si se tuviera un conjunto de datos más grande. De esta forma este trabajo se puede considerar como una continuación, estudiando la posibilidad de incorporar datos sin etiquetas bajo este contexto.

En una primera instancia este trabajo considera modelos de extracción de características solo para dibujos, considerando extender los modelos para el contexto de recuperación de imágenes basada en dibujos. Se decide realizar esto debido a que el dominio de los dibujos presenta características que hacen que trabajar con estos sea más complejo que utilizar fotos. En particular existe una gran variabilidad entre dibujos, incluso en aquellos que representan un mismo objeto, pues entran en juego variables como: la habilidad de la persona que realiza el dibujo, qué ejemplo en particular se elige de cada clase y como la persona interpreta el objeto.

Este trabajo no busca entregar una solución definitiva para el problema, sino realizar una investigación en detalle sobre la viabilidad de modelos de redes neuronales auto-supervisados y semi-supervisados específicos para la extracción de características en el contexto de recuperación de imágenes basada en dibujos. La idea es comparar los modelos a estudiar con los modelos supervisados que se utilizan actualmente, utilizando métricas relevantes según el problema. Finalmente, se espera que este trabajo sea capaz de proponer una arquitectura de red neuronal para recuperación de imágenes basada en dibujos que pueda entrenarse con datos sin etiquetas.

[1] *Impresee CreativeSearch.* `https://impresee.com/shopify-search-bar-and-filters/`

1.3. Objetivos

1.3.1. Objetivo general

Investigar, implementar y evaluar modelos auto-supervisados y semi-supervisados para la extracción de características visuales en el contexto de recuperación de imágenes basadas en dibujos.

1.3.2. Objetivos específicos

1. Definir conjuntos de datos de dibujos para el entrenamiento y evaluación de modelos.
2. Implementar VAE y evaluar su viabilidad como extractor de características de dibujos.
3. Estudiar, diseñar e implementar formas de extender VAE en el contexto semi-supervisado, y evaluar su viabilidad como extractor de características de dibujos.
4. Implementar BYOL y evaluar su viabilidad como extractor de características de dibujos.
5. Definir conjuntos de datos de dibujos e imágenes para el entrenamiento y evaluación de modelos en el contexto de recuperación de imágenes basada en dibujos.
6. Extender BYOL para aprender representaciones tanto de dibujos como de fotos.

2. Marco teórico y estado del arte

2.1. Entrenamiento supervisado, no supervisado y semi-supervisado

En el área de machine learning e inteligencia artificial se utiliza un gran número de datos para entrenar modelos matemáticos de forma que estos puedan aprender a realizar distintas tareas. Existen principalmente dos tipos de entrenamiento: supervisado y no supervisado, en donde el primero corresponde a una metodología más clásica en donde se guía al modelo para que aprenda en la dirección que uno espera, para esto necesita información extra para cada elemento, en general una etiqueta o *label*, que es lo que uno espera que el modelo aprenda. Por otro lado, los métodos no supervisados solo le entregan los datos al modelo y aprenden lo que puedan de estos, y aunque sí se pueden imponer restricciones durante el aprendizaje, estas no fuerzan el aprendizaje hacia una dirección en particular.

Los métodos supervisados, como clasificación o regresión, son los que en general logran mejores resultados en la mayoría de los problemas, y por lo tanto, hasta el momento existen muchos más modelos y estudios sobre esta forma de entrenar. Sin embargo, la desventaja de esto es que se necesita tener información extra de los datos usados para entrenar, por lo que previamente se debe realizar un trabajo de etiquetado o clasificación que en la mayoría de los casos es realizado por personas y puede ser muy costoso. Es debido a esto que existe interés en modelos no supervisados, si bien hoy en día existe una diversa y enorme cantidad de datos, una gran parte de estos no están etiquetados. Existe también un subconjunto de los modelos no supervisados conocidos como auto-supervisados, estos en vez de utilizar una etiqueta son capaces de generar una pseudo etiqueta para cada dato.

Además de los métodos supervisados y no supervisados, existe una versión híbrida conocida como métodos semi-supervisados, que permiten aprender de datos con y sin etiquetas. La idea es que dado un problema, el mejor caso es poder utilizar todos los tipos de datos disponibles, ya sean los datos etiquetados pues entregan información muy valiosa para el modelo y los datos no etiquetados para lograr una mejor generalización. En general, estos modelos necesitan solo una pequeña porción de datos con etiquetas y una gran cantidad sin estas, pues es como realmente se distribuyen los datos en la práctica, debido a esto se espera que tengan mejores resultados que un modelo no supervisado.

2.1.1. Clasificación

Clasificación es uno de los problemas más comunes en el área de *machine learning*, este consiste en aproximar una función f que sea capaz de predecir la clase o etiqueta y de un elemento x. En general, para lograr aproximar esta función se necesita un conjunto de entrenamiento que tenga datos y etiquetas asociadas, que se utilizan para ajustar el modelo. Existen varios algoritmos clásicos para este problema, desde modelos probabilísticos a simples definiciones de distancia entre los datos, sin embargo, hoy en día las redes neuronales constituyen el estado del arte para este tipo de problemas.

2.1.2. Clasificador *K-Nearest Neighbors*

Una forma simple de definir un clasificador es utilizando el algoritmo de *K-Nearest Neighbors*, para aplicar este algoritmo se necesita un conjunto Q de elementos que se necesite clasificar y una muestra C de elementos que sí tengan una clase asignada, luego el algoritmo toma cada $q \in Q$ y le asigna la clase más frecuente entre los $\mathbf{K}$ elementos más cercanos, por ejemplo en la Figura 1 con $\mathbf{K} = 4$ la clase asignada sería "roja". Esta definición de cercanía depende del problema, pero en general se define como la distancia euclidiana entre vectores.

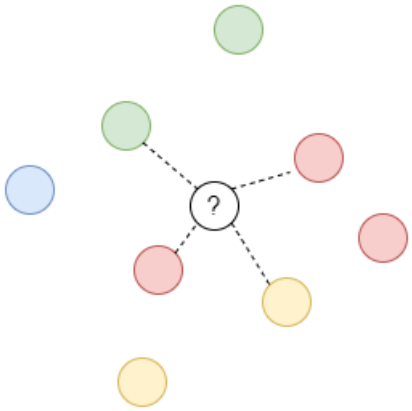


Figura 1: Visualización del algoritmo K-Nearest Neighbors con K = 4, en este caso el elemento central debería tomar la clase roja.

2.1.3. Regresión

En el contexto de *machine learning*, un problema de regresión corresponde a aproximar una función f que dado un dato x sea capaz de producir un valor y, pero a diferencia del clasificación este valor y puede ser un valor continuo en vez de una etiqueta. Los problemas de regresión en general buscan predecir ciertos comportamientos, por ejemplo: predecir la temperatura de un día futuro, predecir cuántos clientes se esperan en cierta fecha, predecir precios con características secundarias, etc. Al igual que en el problema de clasificación, existen varios modelos y algoritmos matemáticos para resolver este problema, incluyendo las redes neuronales.

2.1.4. Redes Neuronales

Una red neuronal es un modelo matemático inspirado en el funcionamiento de las neuronas, en donde se tiene una unidad básica llamada *perceptrón* que al igual que las neuronas recibe múltiples entradas y entrega una sola salida. En la neurona las *dendritas* reciben varios impulsos eléctricos y si el estímulo supera cierto umbral el axón transmite una señal, en un perceptrón en cambio, se recibe un vector de entrada y se realiza un producto punto con un vector de pesos y si el valor supera cierto umbral se entrega un valor mayor a 0. Esto se puede ver en la Figura 2, en donde los x_i componen al vector de entrada, los w_i son los pesos

del perceptrón y f es la función de activación, notar que también se suma un *bias* b_i que permite que el perceptrón pueda aproximar más funciones. El poder de las redes neuronales consiste en poder conectar enormes números de perceptrones, y a través de una operación llamada *backpropagation*, se pueden utilizar los gradientes de los pesos de cada perceptrón para entrenar a la red.

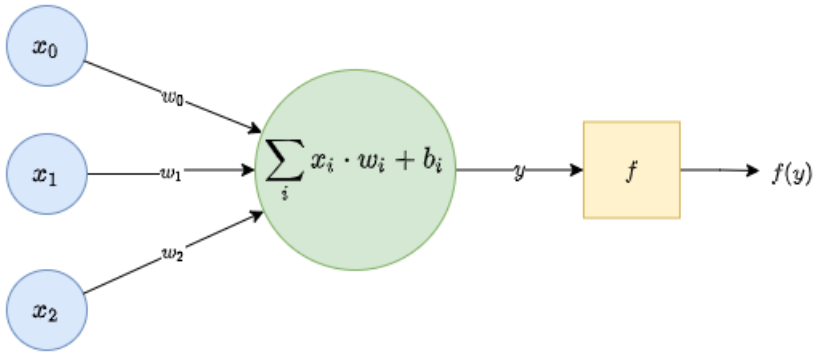


Figura 2: Visualización de un perceptrón, los círculos azules corresponden a las entradas con las que se realiza un producto punto con los pesos del perceptrón y finalmente se aplica una función de activación f a la salida.

Cuando se utiliza un gran número de perceptrones se tiene una función con un gran número de parámetros entrenables, en general se utilizan modelos con millones de parámetros, y con esto es posible aproximar funciones de muy alta complejidad. Si bien las redes neuronales son un concepto que nace alrededor del año 1940, estos modelos solo han ganado protagonismo en las últimas dos décadas. Esto debido que como poseen un alto número de parámetros también necesitan un alto número de datos para el entrenamiento y un alto poder de procesamiento, lo que solo ha sido posible gracias a las nuevas tecnologías.

2.1.4.1 Capas lineales, densas o *fully-connected*

Las primeras redes neuronales consistían de capas de perceptrones, cada capa da como resultado un vector en donde cada componente corresponde a la salida de un perceptrón, estas salidas de cada capa pueden ser usadas como entrada de la capa siguiente hasta terminar con un vector final como salida de la red. Cada perceptrón de una capa recibe como entrada todo el vector de la capa anterior, de esta forma se entiende que cada perceptrón está conectado con todos los perceptrones de la capa anterior y con todos los perceptrones de la capa siguiente, como se ve en la Figura 3. Una capa conectada de esta forma es conocida como una capa *fully-connected*.

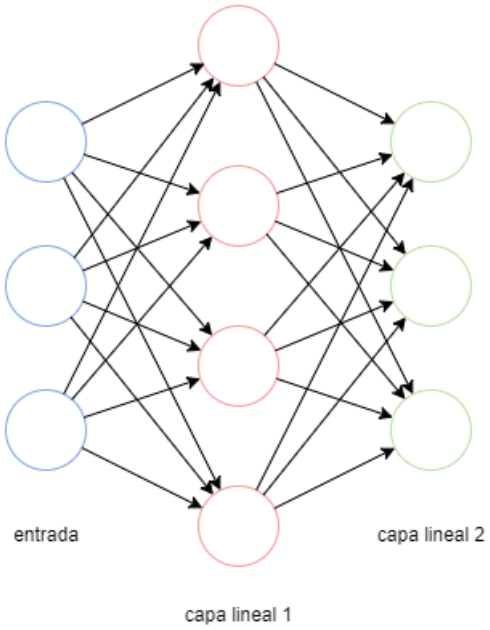


Figura 3: Conexión de capas *fully-connected*, cada color representa una capa distinta.

2.1.4.2 Capas convolucionales

Si bien una capa densa hace sentido cuando se trabaja con vectores, al momento de trabajar con imágenes el equivalente sería conectar el valor de todos los pixeles a cada perceptrón de la capa siguiente. Esto presenta dos problemas: la cantidad de píxeles en una imagen escalan de forma cuadrática con el tamaño de la imagen y se pierde la noción de localidad en la operación. Por esto se define una capa inspirada en la operación de convolución, que en su forma discreta se puede entender como pasar filtro cuadrado más pequeño por la imagen, realizando una multiplicación punto a punto como se ilustra en la Figura 4. Estos filtros presentan además una cantidad de canales de salida, luego la salida de una de estas capas es un cuadrado con dimensiones menores a la entrada y con una tercera dimensión igual a la cantidad de canales del filtro.

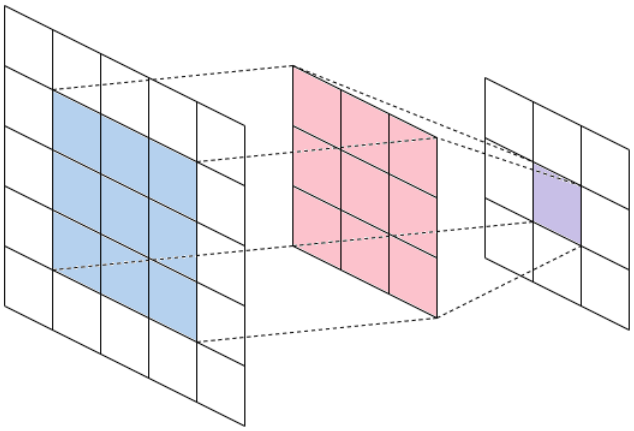


Figura 4: Aplicación de un filtro en capa convolucional, donde el filtro rojo realiza un producto punto con la sección azul, para producir el valor morado en el resultado.

2.1.4.3 Funciones de activación

Para definir el umbral de activación de una capa de red neuronal se utilizan funciones de activación, estas se aplican después de cada capa en la gran mayoría de los casos y ayudan a mejorar el aprendizaje de la red. Existen varias funciones de activación, algunas utilizadas en este trabajo son:

1. **Sigmoid:** Esta función busca representar un umbral duro como una función escalón, pero de forma suave, y por lo tanto derivable. Simplemente empuja los valores superiores a 0 hacia el valor 1 y los valores menores a 0 al valor 0, como se ve en la Figura 5. Al aplicarse esta función a un vector se aplica de forma individual a cada componente. La función se encuentra definida en la Ecuación 1.

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

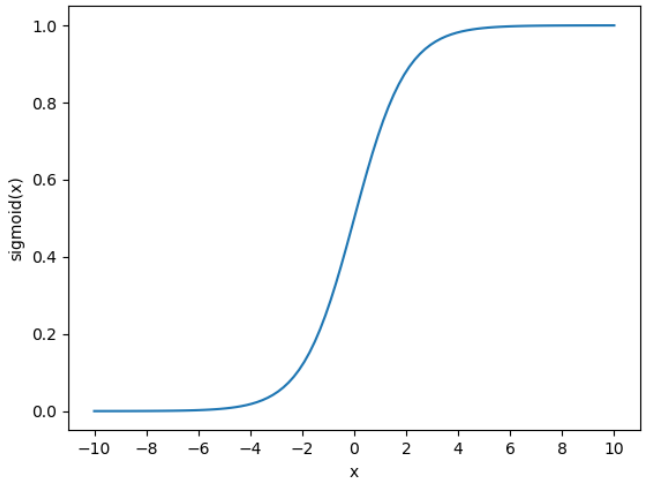


Figura 5: Forma de la función *sigmoid*.

2. **Softmax:** Esta función, a diferencia de *sigmoid*, se debe aplicar a un vector en vez de a cada componente. Como su nombre lo indica busca aplicar la función máximo de forma suave, es decir, al valor máximo del vector lo empuja a 1 y al resto lo empuja hacia cero, como se ve en la Figura 6. Esta función en general se aplica a la capa final en los modelos usados para clasificación. Esta función se define como la expresión de la Ecuación 2, siendo N el largo del vector.

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{2}$$

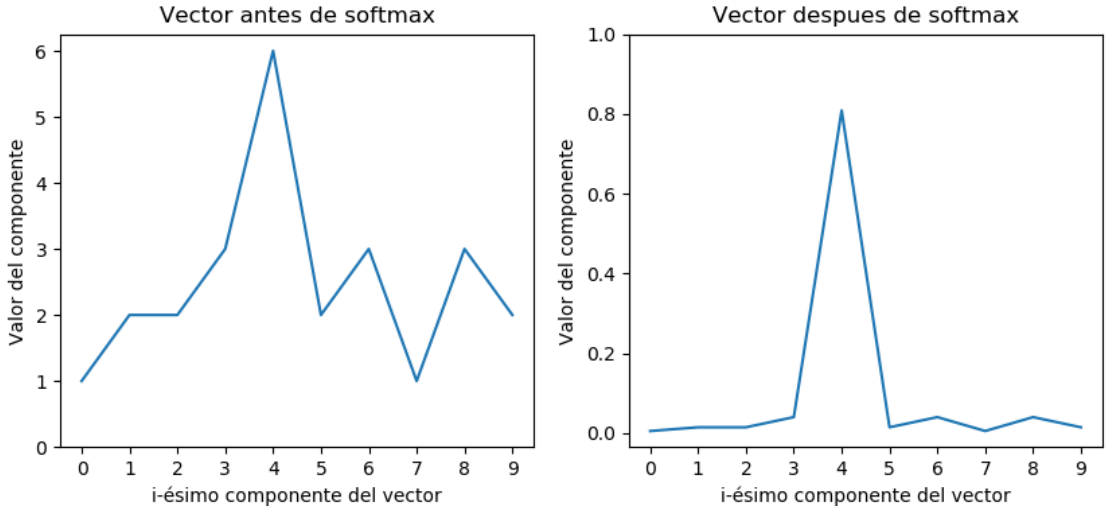


Figura 6: Efecto de la función *softmax*.

3. **ReLU:** Esta función busca representar un umbral pero sin escalar los valores entre 0 y 1. Simplemente define que todos los valores menores a 0 sean igual a 0, mientras que

el resto de los valores permanece igual, como se ve en la Figura 7. Esta función es muy utilizada como función de activación de las capas intermedias, pues si en cambio se utiliza *sigmoid* en redes muy profundas, los valores tienden a 0 al final de la red. La expresión de $ReLU$ se encuentra en la Ecuación 3.

$$\text{ReLU}(x) = \begin{cases} 0 & x \leq 0 \\ x & \text{en otro caso} \end{cases} \tag{3}$$

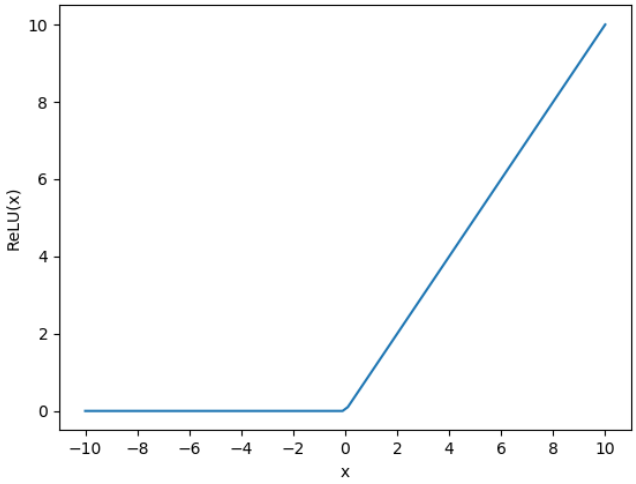


Figura 7: Forma de la función $ReLU$.

2.1.4.4 Funciones de pérdida

Para que una red neuronal aprenda, primero es necesario definir un objetivo, esto se hace a través de la función de pérdida que corresponde a la función que se busca optimizar. En general, se busca una función que cuando su valor sea cero entonces el problema se resuelva de forma perfecta, y luego durante el entrenamiento se minimiza la función de forma progresiva. También hay otros factores importantes a la hora de elegir una función de pérdida, la forma de la función, y en particular de su derivada, afectan la velocidad con la que la red aprende. En este trabajo se utilizan funciones de pérdida conocidas como:

1. **Binary Cross-Entropy:** Esta función se utiliza en problemas donde los resultados son binarios. Para usar esta función es necesario definir ambos estados con los valores 0 y 1, además es común usarla con una función sigmoide como función de activación de la capa final. Esta función se describe en la Ecuación 4, siendo $\hat{y}$ la salida del modelo, y el valor real y N el largo del vector de salida del modelo.

$$\text{BinaryCrossEntropy}(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i) \tag{4}$$

2. **Cross-Entropy:** Esta función es comúnmente utilizada en problemas de clasificación con múltiples clases. Para usar esta función se necesitan vectores de clase en donde cada

componente representa la probabilidad de pertenecer a una clase en específico, la clase real debe ser también un vector con valor uno en el componente que represente a la clase correspondiente y cero en el resto, esto se conoce como *one-hot encoded*. Además se utiliza en conjunto con una función de activación *softmax* en la última capa del modelo. La función se encuentra en la Ecuación 5, con $\hat{y}$ siendo la salida del modelo, y el valor real y N el largo del vector de salida del modelo.

$$\text{CrossEntropy}(y, \hat{y}) = -\sum_{i=1}^{N} y_i \cdot \log \hat{y}_i \tag{5}$$

2.2. Extracción de características en imágenes

La extracción de características o *features* en imágenes busca obtener valores de cada imagen de forma que estos sean capaces de representarlas en un espacio vectorial de baja dimensionalidad. Estos valores pueden describir formas, texturas, colores o cualquier tipo de información que pueda ser extraída de una imagen, luego estas características pueden definir relaciones de similitud más complejas que solo una comparación pixel a pixel. Este problema sigue siendo muy relevante al día de hoy, pues es la base de problemas más complejos como clasificación y regresión, por lo que se siguen investigando nuevas técnicas para obtener buenas representaciones de imágenes.

La forma clásica de extraer características de imágenes se basa en aplicar distintas operaciones matemáticas para obtener valores numéricos, esto puede ir desde obtener histogramas de color a contar los ángulos de los bordes de las figuras. Sin embargo, hoy en día las redes neuronales han pasado a ser el centro de atención en esta área, esta técnica consiste en entrenar redes para luego obtener vectores de características de las salidas de capas intermedias de la red, denominados como *deep feature vectors*, en general se usa la capa previa a la capa que realiza una predicción como se muestra en la Figura 8. La ventaja de usar redes es que no solo aprenden una buena representación sin tener que elegir el mejor método, sino que también presentan mejores resultados que las técnicas tradicionales y han pasado a ser el estado del arte. La desventaja de utilizar redes neuronales es la alta cantidad de datos necesaria y los largos tiempos de entrenamiento.

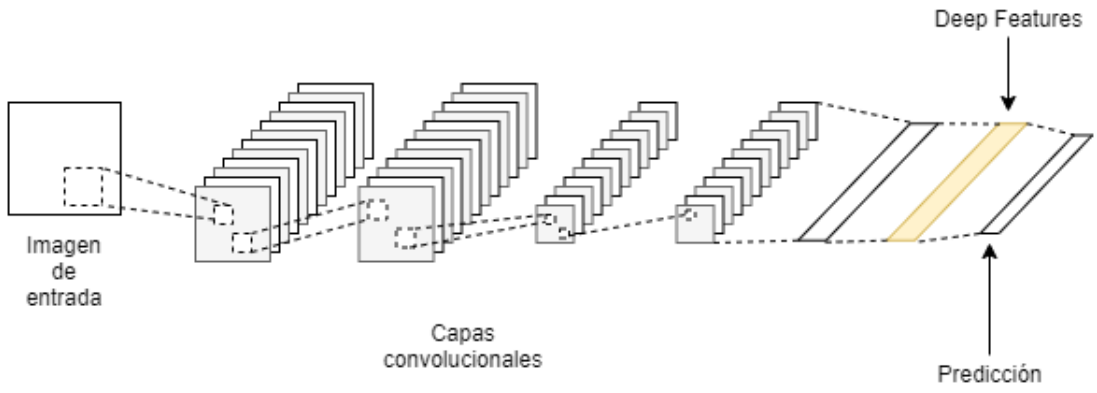


Figura 8: Extracción de deep vectors en redes neuronales.

2.3. Mean Average Precision (mAP)

En los problemas de recuperación de información donde se realizan búsquedas con alguna llave o consulta, una métrica muy popular es Mean Average Precision o mAP, esta busca medir qué tan relevante es la información recuperada y toma valores entre 0 y 1 en donde los valores más altos son mejores. Para calcular esta métrica se necesita un conjunto de queries Q, las consultas que se van a realizar, y un catálogo del cual se recupera un conjunto R de datos, luego se calcula mAP con los siguientes pasos:

1. Para cada $q \in Q$ se define Γ_q como el conjunto de elementos de R que son relevantes para q y una función característica f_{Γ_q} que asigna un valor 1 a los elementos relevantes a la consulta y 0 al resto:

$$f_{\Gamma_q}(x) = \begin{cases} 1 & x \in \Gamma_q \\ 0 & \text{en otro caso} \end{cases} \tag{6}$$

2. Luego se define la precisión del elemento i de R como una función $pr_q(i, R)$, notar que si r_i no es relevante el valor siempre es 0:

$$pr_q(i, R) = \frac{\sum_{k=1}^{i} f_{\Gamma_q}(r_k)}{i} \cdot f_{\Gamma_q}(r_i) \tag{7}$$

3. Se calcula el Average Precision o AP de una consulta q como el promedio de las precisiones de todos los elementos de R:

$$AP_q(R) = \frac{\sum_{i=1}^{|R|} pr_q(i, R)}{|R|} \tag{8}$$

4. Finalmente, se define Mean Average Precision o mAP como la media de todos los Average Precision de cada consulta:

$$mAP(Q, R) = \frac{\sum_{q \in Q} AP_q(R)}{|Q|} \tag{9}$$

Debido a que calcular mAP es caro computacionalmente existen variaciones en donde el cálculo de cada AP está limitado hasta encontrar un número **K** de elementos relevantes, en este caso la notación pasa a ser mAP@K, esto sería cambiar el límite superior de la sumatoria de la Ecuación 8 por **K**.

2.4. Recuperación de imágenes basada en dibujos mediante redes convolucionales

El problema de la recuperación de imágenes basada en dibujos, o *Sketch Based Image Retrieval* (SBIR) en inglés, consiste en realizar una consulta con un dibujo hecho a mano

para encontrar las imágenes reales o fotos más parecidas desde una base de datos o catálogo, se puede ver un ejemplo de búsqueda en la Figura 9. El objetivo es encontrar una manera de extraer características tanto de dibujos como de imágenes de forma que estas puedan ser comparadas, siendo la dificultad que ambos dominios son muy distintos entre sí. Hay mucha información que se puede extraer de una imagen, como contornos, dibujos o texturas, sin embargo en el dominio de los dibujos solo se tienen trazos en un fondo, además la forma en que cada individuo interpreta y expresa un objeto en forma de dibujo varía de persona a persona. Existe un estudio previo [1] sobre este tema en la memoria realizada por Aníbal Fuentes, bajo la guía del profesor José Saavedra, en donde se enfrenta el problema a través de redes convolucionales utilizando tres técnicas.

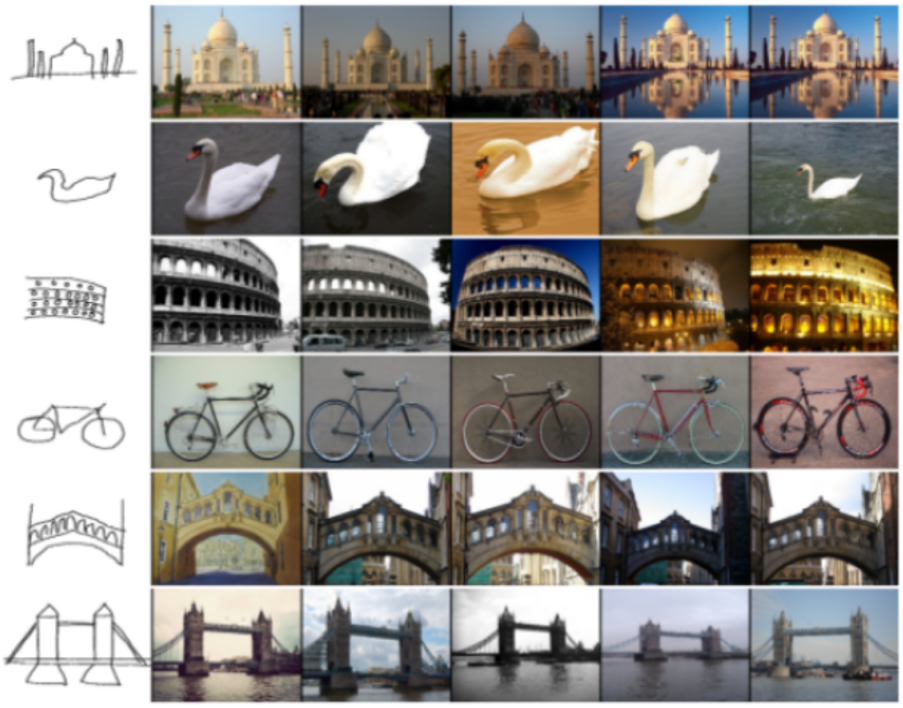


Figura 9: Ejemplo de consultas en SBIR, la primera columna muestra consultas de dibujos y a la derecha de estas las imágenes obtenidas.

2.4.1. Deep-SBIR

En esta técnica [2] se entrenan modelos para el problema de clasificación utilizando dibujos hechos a mano y pseudo dibujos generados a partir de imágenes, a través del algoritmo de Canny [3] para detección de bordes. De esta forma se pueden obtener *deep feature vectors* desde las redes entrenadas, notar que estos modelos nunca trabajan directamente con una imagen sino con los bordes de estas y por lo tanto ambos dominios no son tan distintos. Luego se define la similitud entre elementos usando la distancia euclidiana entre los vectores de características que se obtienen de la red, de forma que mientras más cercanos sean estos más similares son las imágenes. El proceso completo se muestra en la Figura 10.

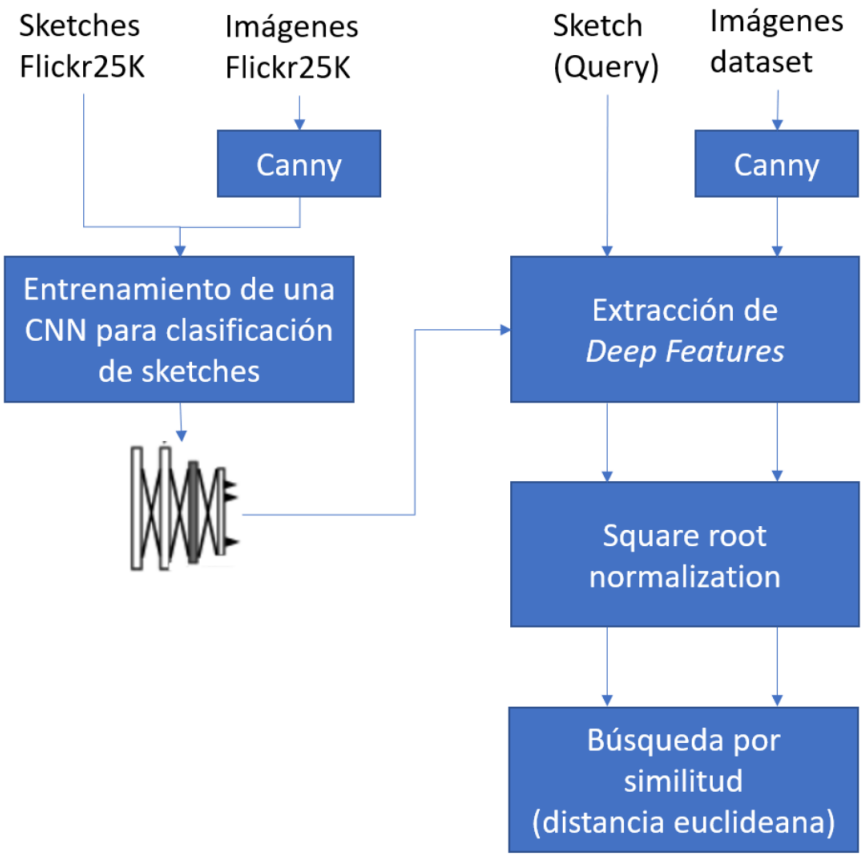


Figura 10: Diagrama de flujo para Deep-SBIR.

2.4.2. SBIR basada en redes siamesas

Las redes siamesas [4] son un tipo de red neuronal en la que se tienen dos o más redes paralelas de arquitecturas idénticas, en donde estas redes pueden o no compartir pesos dependiendo de si reciben entradas del mismo dominio, como se ve en la Figura 11. Estas redes se usan para problemas en donde hay que determinar si dos entradas corresponden a versiones distintas de un mismo objeto, para esto se entrenan con tuplas de elementos del mismo contenido, conocido como pares positivos, y se pasan por las distintas ramas de la red siamesa, luego se utiliza una función de perdida llamada *contrastive loss* que minimiza la distancia entre todas las salidas de estas ramas.

Existe también una mejora de *contrastive loss* en donde además de los pares positivos se utilizan también pares negativos, con contenidos distintos cuyas representaciones deben alejarse. En este caso la red recibe tripletas con un *anchor*, con un positivo y un negativo respecto a este, con esto la función de pérdida pasa a ser un *triplet loss*. Para resolver el problema de SBIR [5] se usan dos ramas en la cual una recibe dibujos y la otra pseudo dibujos generados a partir de imágenes, y se entrena utilizando *triplet loss*. Como ambos dominios son similares los pesos son compartidos entre ramas y finalmente los vectores de características se obtienen de la última capa de estas redes.

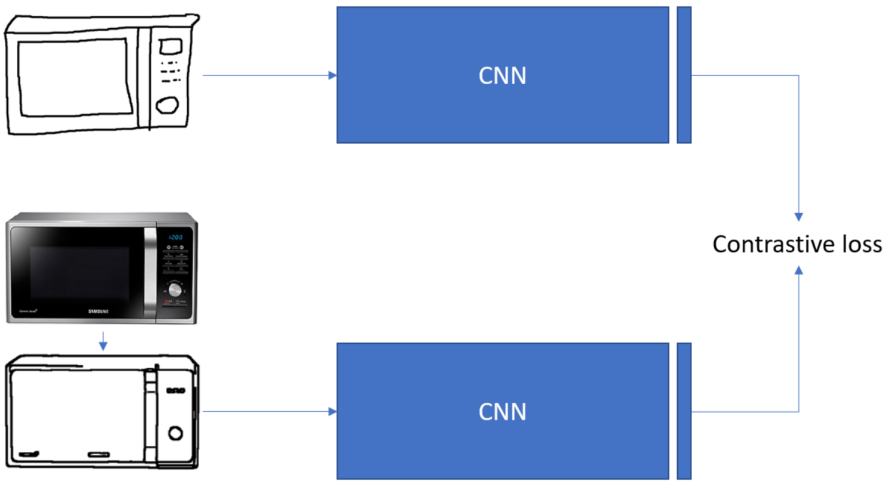


Figura 11: Diagrama de Siamese SBIR.

2.4.3. SBIR basada en múltiples etapas

Este método, de forma similar a una red siamesa, posee dos redes de arquitecturas idénticas, una para dibujos y otra para fotos, aunque en este caso solo se comparten los pesos de las últimas capas pues se utilizan las fotos directamente sin aplicar el algoritmo de Canny. La principal característica de este método es dividir el entrenamiento en varias partes con distintos objetivos, como se ve en la Figura 12, si bien en la publicación original de Bui et al. [6] se tienen más etapas, en este caso solo se usan tres para simplificar la implementación:

1. La primera etapa consiste en entrenar ambas redes a extraer características de forma independiente, cada una es entrenada como un clasificador utilizando simplemente *cross entropy* como función de pérdida.
2. La segunda etapa busca unificar la codificación que realizan ambas redes entrenando las capas finales compartidas, para esto se agrega un *contrastive loss* para acercar las salidas de ambas redes además de las pérdidas de clasificación que se utilizan en la etapa anterior.
3. La tercera etapa corresponde a un ajuste más fino de la red, en donde se integran pares negativos y se utiliza *triplet loss* en conjunto con las pérdidas de clasificación.

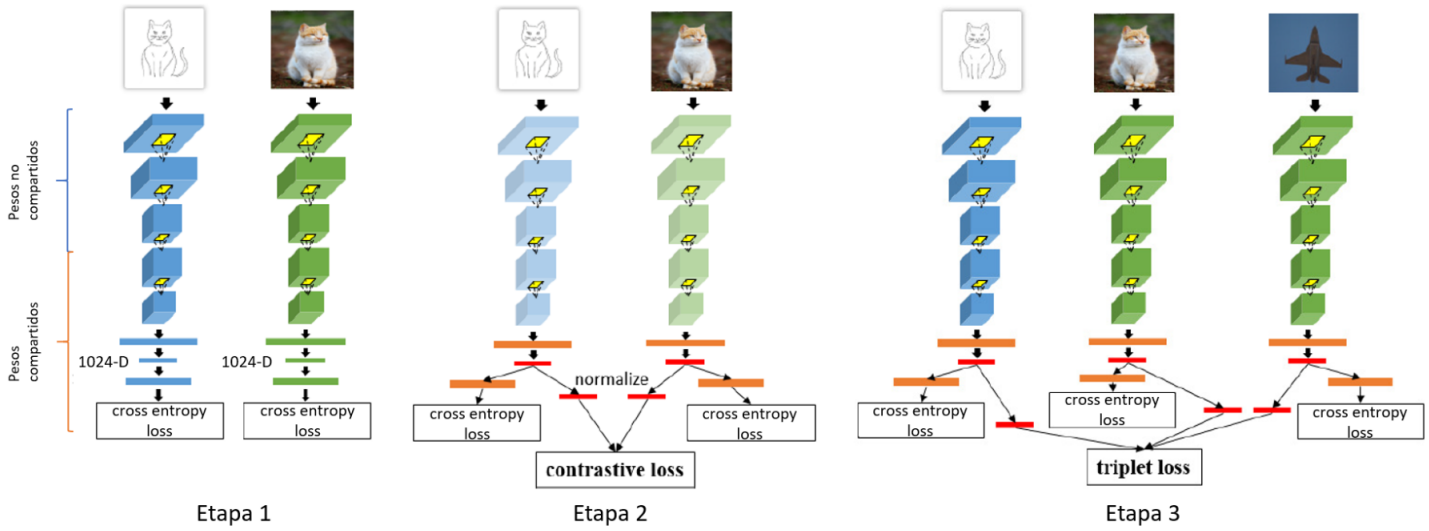


Figura 12: Diagrama con las tres etapas de entrenamiento para Multi Stage Regression SBIR.

2.5. Representaciones efectivas y compactas para recuperación de imágenes basada en dibujos

En el contexto de extracción de características de imágenes, uno de los hiperparámetros que hay que considerar es el tamaño del vector de características. Bajo esta idea, Pablo Torres y José Saavedra realizan un estudio [7] en donde demuestran que utilizar vectores de baja dimensionalidad logra mejores resultados en el problema de recuperación de imágenes basada en dibujos. En este trabajo se utilizan redes basadas en ResNet y se entrenan de forma supervisada para resolver el problema, el tamaño del vector de características que se obtiene desde ResNet es el habitual con 2048 componentes, sin embargo, la técnica consiste en utilizar métodos de reducción de dimensionalidad con el espacio generado.

Existen varias técnicas de reducción de dimensionalidad que se estudian en este trabajo, pero una en particular que muestra tener buenos resultados es el método *Uniform Manifold Approximation and Projection* (UMAP), un método no supervisado que busca reducir la cantidad de dimensiones manteniendo la estructura local del espacio original. Este método se debe aplicar a todo el conjunto de vectores de características para que pueda redistribuirlos en el nuevo espacio de menor dimensionalidad. No se entrenan parámetros que puedan ser utilizados para generalizar a otros datos, sin embargo, el costo computacional de aplicar esta transformación es despreciable cuando el espacio de salida es de baja dimensionalidad. Si se utiliza este método para reducir los vectores a dimensiones muy bajas, en particular 8 y 4 dimensiones, se pudo mejorar la precisión de los modelos utilizados en hasta un 35 %.

2.6. Redes neuronales para imágenes

2.6.1. Arquitecturas de redes neuronales supervisadas relevantes

2.6.1.1 AlexNet

AlexNet [8] es una de las arquitecturas de redes neuronales más famosas e influyentes en el área de visión por computadora, su nombre proviene de uno de los autores del paper original Alex Krizhevsky y fue publicada el año 2012. En este periodo el desempeño de esta red en problemas de clasificación superó de forma considerable a lo que era el estado del arte hasta el momento, al día de hoy esta publicación tiene alrededor de 80000 citas. Esta arquitectura es simple y de esta se basan muchos otros modelos, básicamente consiste de una secuencia de capas convolucionales que van reduciendo las dimensiones de las salidas hasta que cierto punto se aplana todo a vectores, se termina con capas lineales y una función de activación *softmax*, como se ve en la Figura 13, posee además capas de normalización entre medio. AlexNet es un modelo diseñado para ser entrenado de forma supervisada para clasificación de imágenes y es capaz de entregar muy buenos resultados en problemas simples.

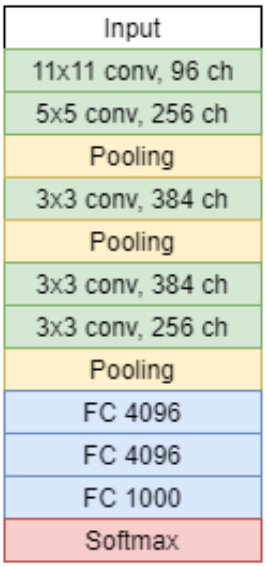


Figura 13: Arquitectura de AlexNet.

2.6.1.2 ResNet

Uno de los problemas que se pueden encontrar en redes neuronales convencionales, incluyendo AlexNet, es el desvanecimiento del gradiente, esto sucede en redes muy profundas en donde los valores de los gradientes de los pesos de la red eventualmente toman valores muy pequeños de manera que el aprendizaje se estanca. ResNet o Residual Neural Network [9] busca solucionar el desvanecimiento del gradiente, permitiendo redes mucho más profundas, y por lo tanto, con mayor capacidad de aprendizaje. Esto lo hace conectando capas de la

red realizando saltos o "skips" en donde el resultado de una capa, además de ser la entrada de la siguiente, se vuelve a reutilizar en una capa posterior en lo que se denomina como un bloque de ResNet como se ve en la Figura 14. ResNet en general se entrena de forma supervisada para problemas de clasificación, pero además de esto se puede usar como extractor de características de imágenes.

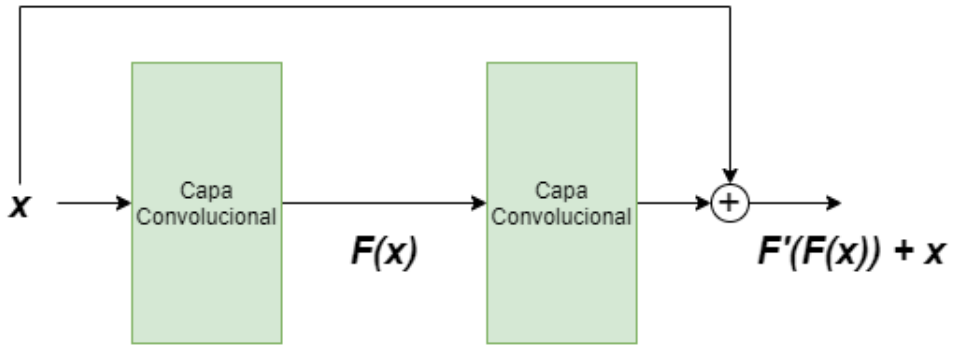


Figura 14: Idea general de un bloque de ResNet, en donde la entrada **x** vuelve a ser utilizada a través de una suma en una capa posterior.

2.6.2. Arquitecturas de redes neuronales auto-supervisadas relevantes

2.6.2.1 Autoencoder

Un Autoencoder es uno de los modelos más básicos para lograr un entrenamiento auto-supervisado, es una arquitectura que posee dos componentes: un *encoder* y un *decoder*. El *encoder* consiste de una red neuronal que recibe la entrada y entrega un vector de salida, la idea es que este vector tenga menos valores que la entrada de forma que sea una codificación de esta. Por otro lado, el *decoder* recibe estos vectores y debe entregar una salida con las mismas dimensiones que la entrada, la idea es que sea capaz de interpretar la codificación para poder reconstruir la entrada, como se observa en la Figura 15. Durante el entrenamiento se pasa un elemento por la red, primero por el *encoder* y luego por el *decoder*, y se usa un función de pérdida que exija que la "reconstrucción" obtenida sea igual a la entrada, de esta forma el modelo es auto-supervisado pues la etiqueta utilizada es la misma entrada.

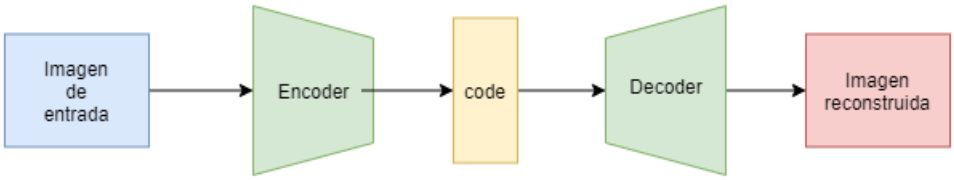


Figura 15: Arquitectura de un Autoencoder.

2.6.2.2 Variational Autoencoder

Los Variational Autoencoders o VAE [10] son un modelo propuesto por Diederik P. Kingma et al., son una mejora sobre los Autoencoder regulares que incorporan distribuciones probabilísticas en el modelo. La razón por la que no basta con utilizar un Autoencoder regular es que estos tienden a simplemente asignar un vector o punto en el espacio a cada imagen, por lo que puede no realmente estar extrayendo características de estas. Un VAE tiene la misma estructura que un Autoencoder, pero el *encoder* en vez de asignar solo un vector a cada elemento, le asigna una distribución de probabilidades, produciendo una media y una varianza para cada uno. Actualmente hay estudios que utilizan esta arquitectura para extraer características de fotos [11].

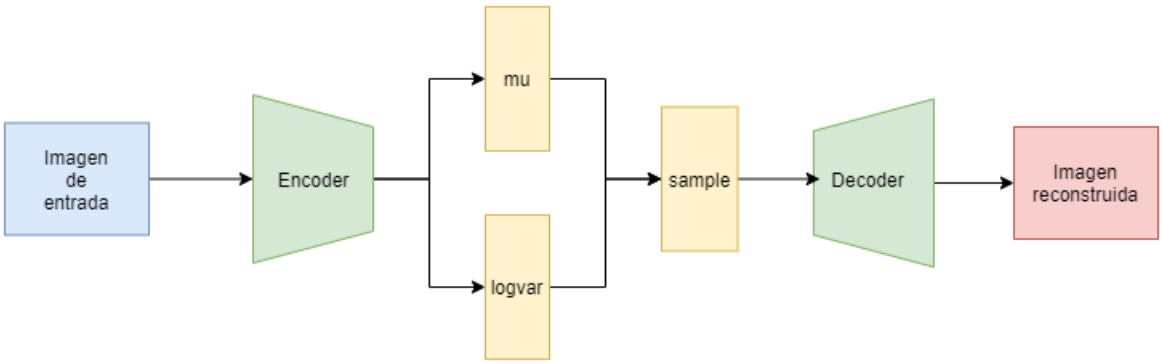


Figura 16: Arquitectura de un Variational Autoencoder, *mu* y *logvar* representan distribuciones de probabilidad.

Los VAE evitan el problema anterior aprendiendo una distribución normal definida por dos vectores *mu* y *logvar* para cada ejemplo, pero al momento de pasar por el *decoder* no se pasan ambos vectores, sino que se obtiene un punto aleatorio de la distribución aprendida en una operación que el autor denomina como *sampling*. Esta operación se puede entender como tomar el vector *mu*, la media de la distribución, y sumarle una perturbación proporcional al vector *logvar*, este ruido evita que la red se sobre ajuste y la obliga a extraer características relevantes. Esta operación de *sampling* solo se realiza durante el entrenamiento, en la evaluación solo se considera el vector *mu* de cada elemento.

$$sample = mu + \mathcal{N}(0,1) \times logvar \tag{10}$$

Otro punto interesante de los VAE es la función de pérdida utilizada, esta posee dos partes en donde la primera corresponde a la pérdida de reconstrucción que se encuentra en cualquier Autoencoder, y se encarga que el modelo sea capaz de reconstruir las imágenes de entrada. El segundo componente corresponde a una divergencia de Kullback-Leibler o KLD, esta métrica mide la diferencia entre dos distribuciones de probabilidad, y en este caso se minimiza la diferencia entre las distribuciones aprendidas y una distribución normal con media 0 y una desviación estándar 1. Esta segunda parte de la función de pérdida regulariza el espacio latente y provoca que los elementos se distribuyan bien dentro de este. Hay estudios [12] que

muestran mejoras en el aprendizaje si se agrega un peso β al componente KLD.

$$\text{VAE Loss} = \text{Recon Loss}(original, recon) + \beta \cdot \text{KLD}(\mathcal{N}(mu, logvar), \mathcal{N}(0, 1)) \tag{11}$$

2.6.2.3 Bootstrap Your Own Latent

Bootstrap Your Own Latent [13] o BYOL es una arquitectura diseñada para aprender a extraer características de imágenes de manera auto-supervisada. Es un modelo reciente, publicado a finales del 2020, que promete resultados competitivos en comparación a las técnicas supervisadas. BYOL es un modelo discriminativo, que a diferencia de los modelos generativos como VAE, solo busca obtener características que permitan determinar si dos elementos pertenecen a la misma clase. La idea principal detras de BYOL es usar dos redes neuronales con una arquitectura casi idéntica, una *online network* y una *target network*, de forma que la primera busca predecir la salida de la segunda durante el entrenamiento, generando así su propia etiqueta.

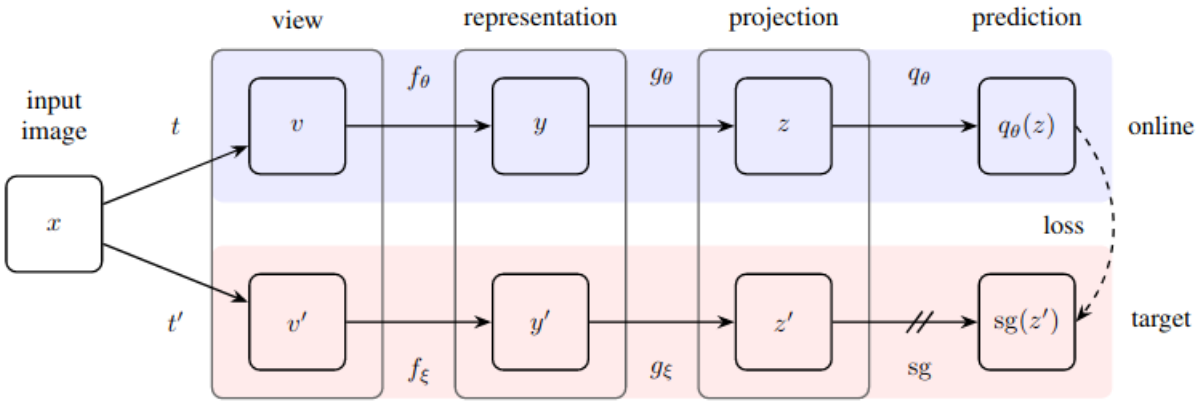


Figura 17: Arquitectura de modelo BYOL, la rama superior corresponde a la *online network*, mientras que la rama inferior corresponde a la *target network*, las características se extraen del componente y.

Tanto la *online network* como la *target network* poseen una arquitectura que consiste principalmente de un *encoder*, que se encarga de efectivamente extraer las características de las imágenes, estos se ven como y e y' en la Figura 17. Además, ambas ramas de BYOL poseen un componente simple con pocas capas llamado *projector*, que se encarga de reducir los vectores de características obtenidos de los *encoder* a un espacio de menos dimensiones antes de realizar la predicción, estos son z y z' en la Figura 17. Finalmente, para la predicción se utiliza un componente más, aunque solo en la rama *online*, llamado *predictor*, representado por q_θ en la Figura 17, que utiliza la proyección z de la *online network* para predecir la proyección z' de la *target network*. Notar que se corta el gradiente después de la proyección z', esto se hace para que la *target network* no sea actualizada al realizar un descenso estocástico del gradiente.

Cada imagen pasa simultáneamente por ambas partes de la red y la forma en que estas interactúan es que la *target network* es una versión más estable de la *online network*, por lo

que durante el entrenamiento la segunda trata de predecir la salida de la primera. Ambas redes son entrenadas en cada pasada, pero se dice que la *target network* es una versión estable pues no aprende a través de un descenso estocástico del gradiente, en cambio, los pesos ξ de esta red se actualizan con un *exponential moving average* desde los pesos θ de la *online network* como se muestra en la Ecuación 12. El parámetro τ indica en qué razón se actualiza la *target network* y toma valores entre 0 y 1, en el caso de BYOL toma parámetros muy cercanos a 1 y por lo tanto aprende de forma muy lenta.

$$\xi \leftarrow \tau\xi + (1-\tau)\theta \tag{12}$$

Además de usar dos redes que aprenden de forma distinta, BYOL requiere que la imágenes que pasan en cada una sean distintas vistas del mismo contenido, y la forma auto-supervisada de lograr esto es aplicando dos transformaciones diferentes a cada imagen que pasa por la red para obtener un par. Esto se ve en la Figura 17, en donde se utilizan las transformaciones t y t' en la entrada x para obtener dos nuevas imágenes v y v'. Al pasar un par de imágenes por la red se hace en ambos sentidos, se predicen mutuamente, y la función de pérdida consiste simplemente de una norma $L2$ al cuadrado, Ecuación 13, una métrica de distancia entre vectores que busca acercar las salidas de ambas redes. Aunque ya existen otros modelos que acercan pares de imágenes de la misma clase, en general estos también requieren pares negativos para alejar objetos en el espacio latente para que este espacio no colapse, situación en que la red termina dando el mismo vector de características para todos los elementos, pero BYOL se destaca en que solo necesita pares positivos pues la *target network* y su aprendizaje lento evitan este problema.

$$\text{SquaredL2Norm}(X,Y) = ||X-Y||_2^2 = 2 - 2\frac{X \cdot Y}{||X||_2 \, ||Y||_2} \tag{13}$$

2.6.3. Arquitecturas de redes neuronales semi-supervisadas relevantes

2.6.3.1 Modelo generativo semi-supervisado M2

El mismo año en que Diederik P. Kingma realizó su publicación sobre Variational Autoencoders también publica varias arquitecturas que buscan extender el modelo VAE para poder ser entrenado de forma semi-supervisada. Una de estas es el modelo M2 [14], un modelo que busca separar dos conceptos: la "clase" de la imagen, es decir lo que la imagen quiere representar y el "estilo" de la imagen, que corresponde características más abstractas que se refieren a la forma en que se representa dicha clase. De esta forma se tiene un *encoder* que se encarga de extraer el estilo de la imagen y en paralelo un clasificador usado para inferir la clase de la entrada, luego se tiene un *decoder* que recibe tanto el estilo inferido por el *encoder* y la clase inferida por el clasificador para poder reconstruir la entrada, esto se puede ver en la Figura 18. En este caso el vector de características que se obtiene es la concatenación del vector de estilo y del vector de clase.

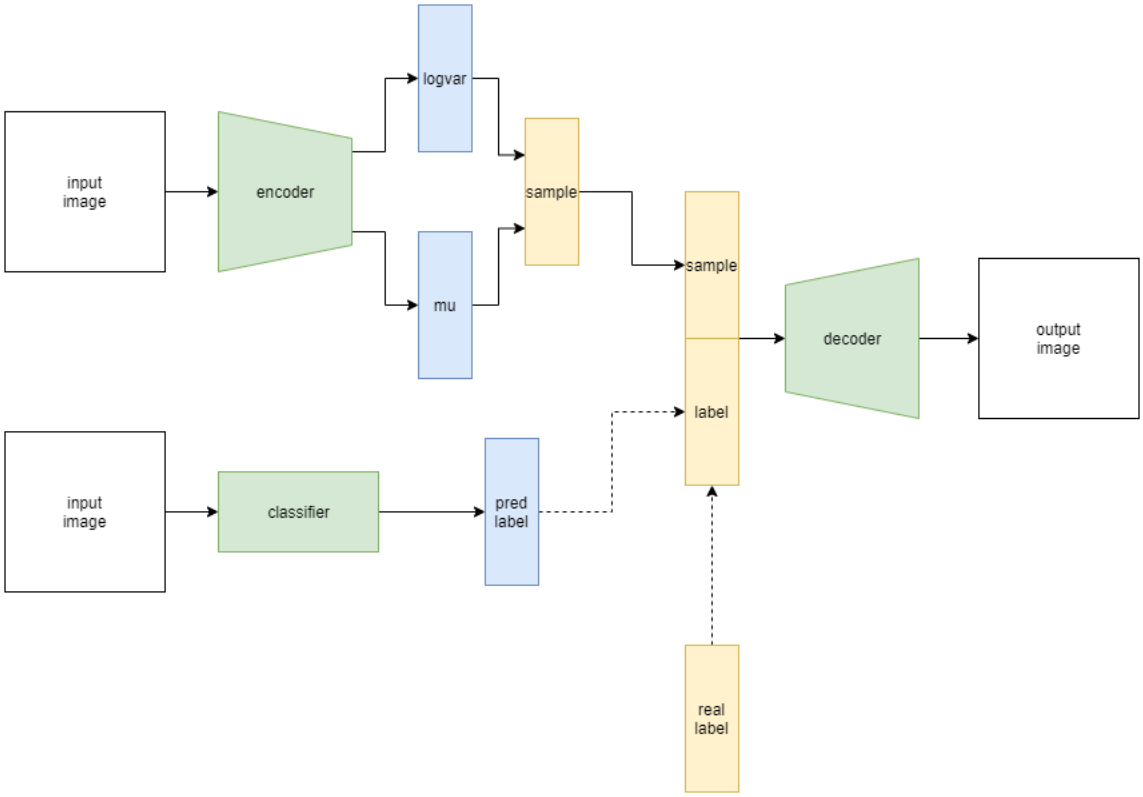


Figura 18: Arquitectura de modelo M2, consiste de una parte similar a un VAE que produce un *sample* y a un clasificador, el *decoder* utiliza la salida de ambas partes para reconstruir la imagen.

El modelo M2, al ser semi-supervisado, es capaz de recibir como entrada imágenes con y sin una clase asociada, y durante el entrenamiento se comporta de forma distinta según cual sea el caso. Cuando se tiene un par de imagen y clase, se usa el *encoder* para obtener el estilo, pero como se tiene la clase simplemente se pasa esta como *one-hot encoded*, luego se concatenan ambos vectores, y a partir de este nuevo vector se reconstruye la imagen con el *decoder*. En los casos en que solo se tiene la imagen se realiza lo mismo, pero se utiliza la salida del clasificador para obtener una clase inferida, notar que esta salida incluye un función de activación *softmax* de forma que representa la probabilidad de pertenecer a cada clase.

De la misma forma, la función de pérdida descrita para el M2 también se divide en dos partes. En el caso supervisado se considera la misma función de pérdida de un VAE, que en este caso se denomina $\mathcal{L}$, y se agrega una perdida de clasificación con un peso α. En el caso no supervisado se realiza una sumatoria de $\mathcal{L}$ con respecto al vector de clase inferido, y además se suma la entropía $\mathcal{H}$ de este mismo vector de clase, incentivando predicciones con una clase predominante. Es importante notar que en ambos casos tanto el *encoder* como el clasificador son entrenados.

$$\mathcal{L} = \text{Recon Loss}(original, recon) + \beta \cdot \text{KLD}(\mathcal{N}(mu, logvar), \mathcal{N}(0, 1)) \tag{14}$$

$$\text{Labeled loss} = \mathcal{L} + \alpha \cdot \text{Cross Entropy}(y\ true,\ y\ pred) \tag{15}$$

$$\text{Unlabeled loss} = \sum y\ pred \cdot \mathcal{L} + \mathcal{H}(y\ pred) \tag{16}$$

3. Desarrollo

En este trabajo se busca entrenar modelos de redes neuronales para la extracción de características de dibujos y evaluar su desempeño para realizar búsquedas por similitud. Para esto se definen conjuntos de consultas, y catálogos con los elementos a recuperar, luego se extraen vectores de características de los ejemplos de ambos conjuntos, utilizando los modelos entrenados, y se procede a realizar búsquedas por similitud, el proceso descrito se puede observar en la Figura 19. La búsqueda por similitud consiste en encontrar los elementos del catálogo cuyos vectores de características sean más cercanos al vector de características de una consulta utilizando una distancia euclidiana. La mayoría de los modelos estudiados son entrenados solo para la extracción de características de dibujos, y por esto tanto las consultas como el catálogo consisten solo de dibujos. El último modelo estudiado es entrenado con fotos y dibujos para enfrentar el problema de recuperación de imágenes basada en dibujos, y por lo tanto las consultas son dibujos y el catálogo consiste de fotos.

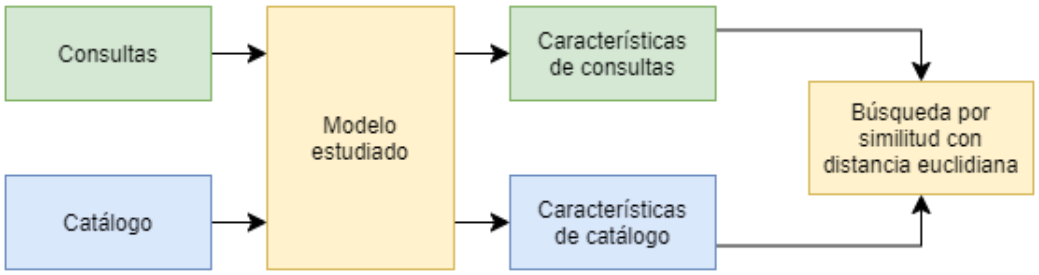


Figura 19: Diagrama de flujo de la búsqueda realizada en los experimentos.

3.1. Modelos basados en VAE

3.1.1. VAE

Con el fin de evaluar un método auto-supervisado para la extracción de características de dibujos se decide estudiar el espacio latente producido por un modelo VAE. En la implementación se consideran imágenes de tamaño 256×256 pixeles, para la arquitectura del *encoder* y del *decoder* se deben tener las siguientes consideraciones: el *encoder* debe ser un modelo que reciba una imagen y dé como resultado un vector, el decoder debe realizar el procedimiento inverso recibiendo un vector y entregando una imagen, la única restricción es que ambas arquitecturas deben tener una capacidad de aprendizaje similar. Para esto se decide utilizar arquitecturas basadas en ResNet 50, un modelo común para problemas de clasificación, de esta forma también se puede usar un ResNet 50 entrenado de forma supervisada para tener una base con la que comparar. En el caso del *decoder* es necesario construir un ResNet 50 invertido, esto se puede lograr fácilmente utilizando capas convolucionales transpuestas, que se pueden entender como la operación inversa a una capa convolucional regular, también teniendo una cantidad de pesos entrenables bastante parecida. La arquitectura usada en los experimentos se puede ver en la Figura 20.

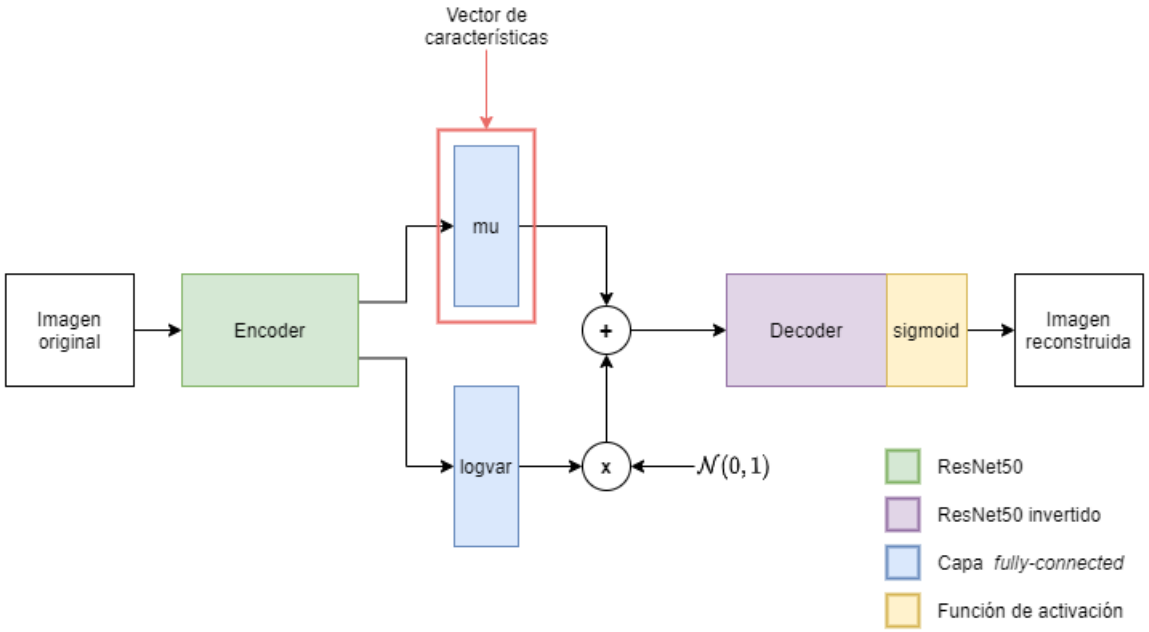


Figura 20: Arquitectura de VAE utilizada en los experimentos, los vectores de características utilizados corresponden a los vectores *mu* que produce el modelo.

Para la función de pérdida es necesario elegir una pérdida de reconstrucción que sea razonable para el dominio de dibujos con el que se está trabajando. Considerando que un dibujo solo tiene dos tipos de pixeles, fondo o trazo, se utiliza *binary cross-entropy*, una función que se usa específicamente cuando se tienen dos estados. Luego se define que los pixeles del fondo del dibujo tienen valor 1, mientras que los que pertenecen al trazo toman valor 0. El utilizar *binary cross-entropy* también implica que la función de activación de la capa final sea una función sigmoide, que tiene la propiedad de empujar los valores hacia 0 o 1. Además como la salida se ha definido como una imagen de valores entre 0 y 1, se debe también escalar los dibujos de entrada para que sean de la misma forma y se pueda aplicar la función de pérdida correctamente. La función de pérdida utilizada es la que se muestra en la Ecuación 17, donde el parámetro β corresponde al peso de la divergencia de *Kullback-Leibler*.

$$\text{Unlabeled loss} = \text{BinaryCrossEntropy}(orignal, recon) + \beta \cdot \text{KLD}(mu, logvar) \tag{17}$$

3.1.2. Modelo M2 semi-supervisado

Con el fin de evaluar la posibilidad de incorporar datos etiquetados a un VAE, se implementa el modelo M2 propuesto por Kingma et al. y se entrena para reconstruir dibujos de forma semi-supervisada. Este modelo separa la información de cada dibujo en dos vectores, uno que representa la clase y otro que representa el estilo del dibujo. Debido a que ambos aspectos son relevantes al momento de realizar una búsqueda, se decide utilizar la concatenación de ambos vectores como vector de características, de la misma forma que el modelo utiliza la concatenación para la reconstrucción. Se diseñan los modelos contemplando imágenes de tamaño 256×256 pixeles. Para las arquitecturas se decide utilizar AlexNet, esto

debido a que en los experimentos se entrenan modelos supervisados con pocos dibujos como una cota inferior con la cual comparar los resultados, y ResNet 50 se sobre ajusta con tan pocos datos. De esta forma tanto el *encoder* como el clasificador corresponden a modelos AlexNet, mientras que el *decoder* corresponde a un modelo AlexNet invertido que utiliza capas convolucionales transpuestas. Esta arquitectura se puede ver en la Figura 21.

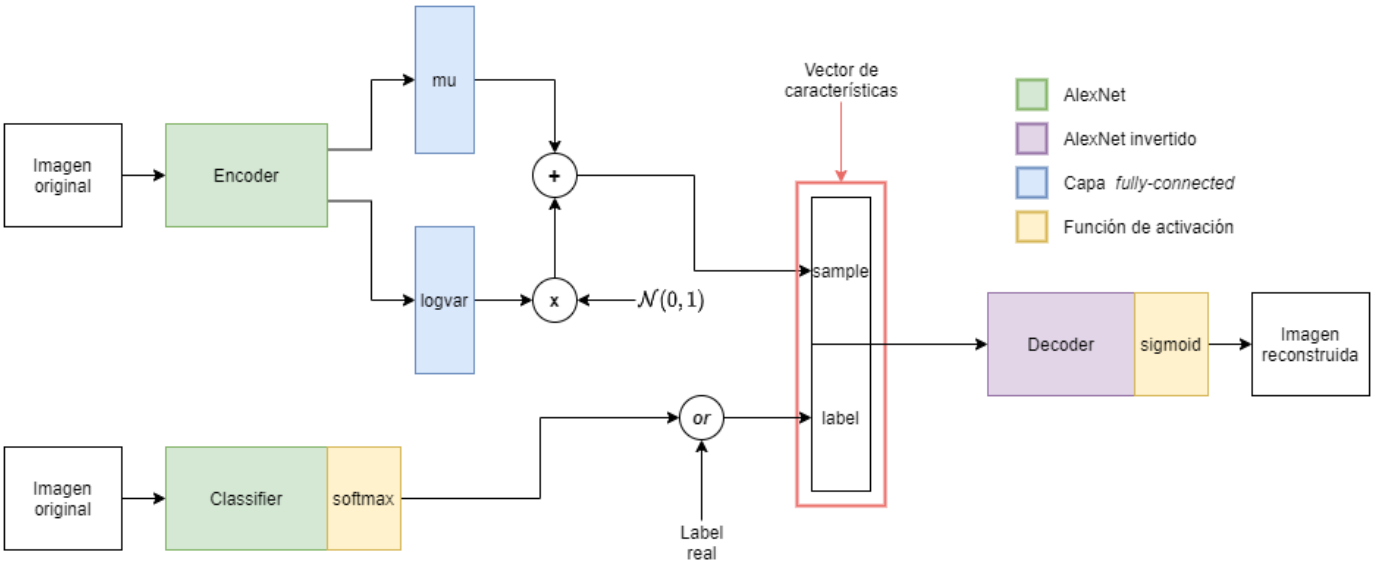


Figura 21: Arquitectura de modelo M2 utilizada en los experimentos, los vectores de características utilizados corresponden a la concatenación de los vectores *sample* que produce el modelo y la etiqueta de cada elemento representada como vector *one-hot encoded*.

Se utiliza *binary cross-entropy* como perdida de reconstrucción y por lo tanto los valores son escalados entre 0 y 1, siendo 0 parte del trazo y 1 el fondo. Se define como $\mathcal{L}$ a la pérdida de un VAE regular, teniendo también un peso β para el componente de KLD como se muestra en la Ecuación 18, este valor es utilizado tanto para la función de pérdida del caso etiquetado como no etiquetado. El caso etiquetado corresponde a la suma de $\mathcal{L}$ con un error de clasificación, como se ve en la Ecuación 19 este error tienen un peso igual a $0{,}1 \cdot N$, siendo N el tamaño del conjunto de datos, como se indica en la publicación original. La pérdida del caso no etiquetado es más compleja, como se puede ver en la Ecuación 20, el valor $\mathcal{L}$ se multiplica con cada componente de la etiqueta predicha por el clasificador y se suman los valores, esto le da más importancia a $\mathcal{L}$ cuando hay predicciones con valores altos. Además se calcula la entropía $\mathcal{H}$ del de la etiqueta predicha, lo que empuja a la red a hacer predicciones hacia solo una clase.

$$\mathcal{L} = \text{BinaryCrossEntropy}(original, recon) + \beta \cdot \text{KLD}(mu, logvar) \tag{18}$$

$$\text{Labeled loss} = \mathcal{L} + 0{,}1 \cdot N \cdot \text{CrossEntropy}(y\ true,\ y\ pred) \tag{19}$$

$$\text{Unlabeled loss} = \sum y\ pred \cdot \mathcal{L} + \mathcal{H}(y\ pred) \tag{20}$$

3.1.3. VAE semi-supervisado

Una forma simple de lograr un VAE semi-supervisado es incorporar un clasificador, de esta manera se espera que el espacio latente conserve las propiedades de uno generado por un VAE, pero aprovechando las características particulares de cada clase. La idea consiste en tener una arquitectura en forma de **Y**, esto se puede apreciar mejor en la Figura 22, en donde cada *sample* obtenido desde el *encoder* se utilice como entrada del *decoder* y del clasificador. En la implementación se consideran imágenes de entrada de tamaño 256×256 pixeles. Con el mismo razonamiento del modelo M2, y para además comparar los resultados semi-supervisados, se utiliza la arquitectura Alexnet para el *encoder* y AlexNet invertido para el *decoder*. El clasificador en cambio no necesita ser una arquitectura muy grande, consiste simplemente de una capa densa o *fully-connected* con un tamaño igual al numero de clases y una función de activación *softmax*.

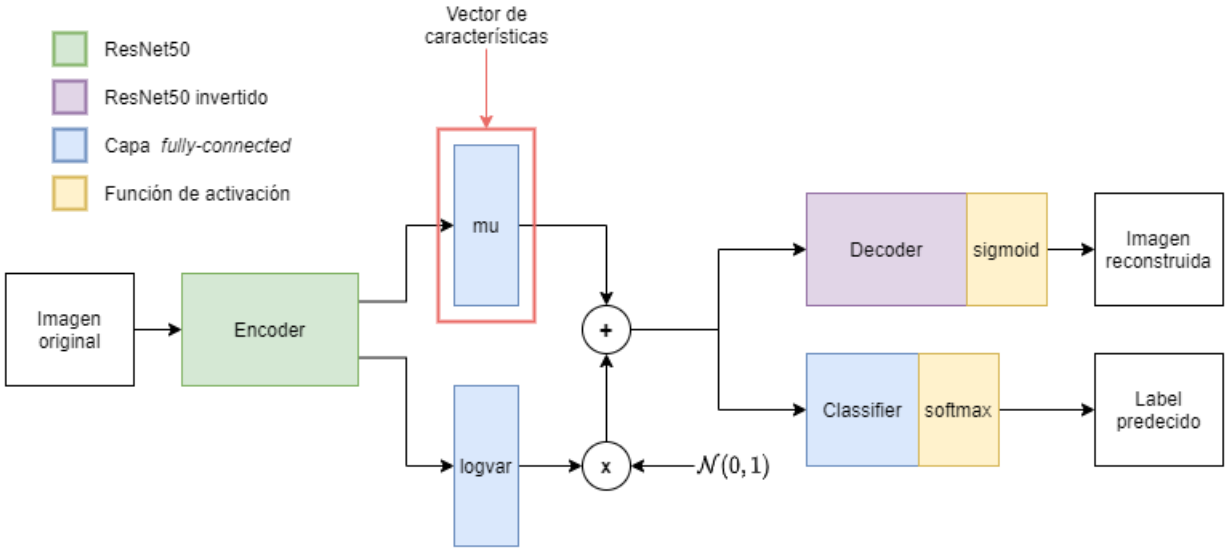


Figura 22: Arquitectura de VAE semi-supervisado utilizada en los experimentos, los vectores de características utilizados corresponden los vectores *mu* que produce el modelo.

La función de perdida entonces consistiría de la misma función de perdida de un VAE más una perdida de clasificación con peso igual a 0,1, este valor debe ser bajo pues valores más altos que esto provocan que la red pase a ser más un clasificador que un VAE, esto se muestra en la Ecuación 21. En el caso no supervisado esta arquitectura se comporta como un VAE regular, teniendo también un peso β para la pérdida de KLD, esto se ve en la Ecuación 22. Para la perdida de reconstrucción se utiliza *binary cross-entropy* y para la perdida de clasificación *categorical cross-entropy*. También los valores son escalados entre 0 y 1, definiendo 0 como parte del trazo y 1 el fondo.

$$\begin{aligned}\text{Labeled loss} = \text{BinaryCrossEntropy}(orignal, recon) + \beta \cdot \text{KLD}(mu, logvar) + \\ 0{,}1 \cdot \text{Cross Entropy}(y\ true, y\ pred)\end{aligned} \tag{21}$$

$$\text{Unlabeled loss} = \text{BinaryCrossEntropy}(orignal, recon) + \beta \cdot \text{KLD}(mu, logvar) \tag{22}$$

3.2. Modelos basados en BYOL

3.2.1. BYOL

Se busca evaluar el modelo BYOL para extracción de características de dibujos. En este modelo los vectores de características se obtienen desde la red online, mientras que el resto de la arquitectura se descarta durante la evaluación. En la publicación original de BYOL se obtienen resultados competitivos en comparación a métodos supervisados en el contexto de fotos, se busca estudiar si el modelo es capaz de obtener resultados similares al utilizar dibujos, teniendo en cuenta que estos poseen características abstractas que dependen de las personas quienes los dibujaron. La ventaja del modelo es no necesitar de etiquetas, pero también es interesante probar modelos que sean discriminativos, en vez de generativos como la familia de los VAE, pues existe la posibilidad de que exigir que el vector de características tenga toda la información necesaria para reconstruir los dibujos afecte la calidad de estos.

Para el modelo se consideran dibujos de tamaño $224 \times 224 \times 3$. En cuanto a la arquitectura del modelo, para poder realizar la operación de *exponential moving average*, en donde se transfiere una pequeña parte de lo aprendido por la *online network* a la *target network*, es necesario que ambas posean la misma arquitectura. Debido a esto, y para comparar los resultados con los obtenidos previamente, se decide utilizar ResNet 50 para ambos casos. Para el resto de los componentes, ambos proyectores y el predictor, se utilizan arquitecturas de *Multi Layer Pereceptron* simples, que corresponden a una capa densa o *fully-connected* que reduce a un vector de 256 dimensiones seguido de una capa de normalización y una capa de activación $ReLU$. La arquitectura completa se muestra en la Figura 23.

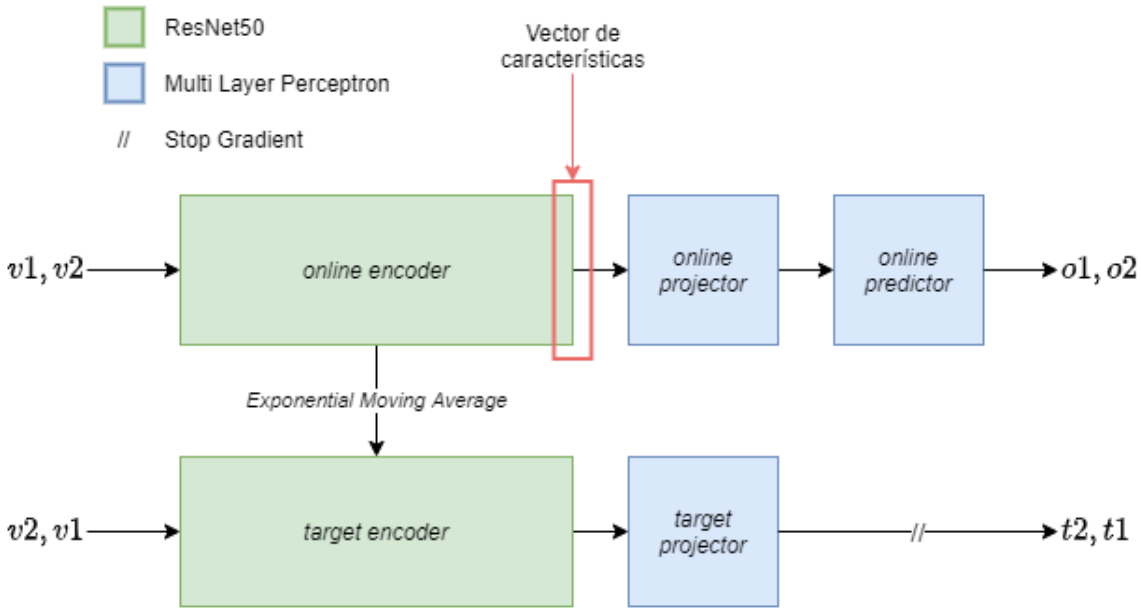


Figura 23: Arquitectura de BYOL utilizada en los experimentos con dibujos, $v1$ y $v2$ corresponden a distintas transformaciones o "vistas" de la imagen original y pasan de forma cruzada por la red. Los vectores $o1$ y $o2$ corresponden a la salida de la *online network*, y $t1$ y $t2$ corresponden a la salida de la *target network*.

La función de pérdida del modelo se utiliza tal cual como se describe en la publicación, usando una norma $L2$ al cuadrado que busca acercar las salidas de ambas redes, esto se hace dos veces pues ambas transformaciones se cruzan en las redes. Siendo $v1$ y $v2$ las dos transformaciones del dibujo, $o1$ y $o2$ las salidas de la *online network* y $t1$ y $t2$ las salidas de la *target network*, la función de perdida final se describe en la Ecuación 23.

$$\text{BYOL Loss} = -\text{SquaredL2Norm}(t1, o2) - \text{SquaredL2Norm}(t2, o1) \tag{23}$$

También, para que BYOL funcione, es muy importante que las imágenes que pasen por ambas ramas sean transformaciones distintas de una misma imagen original, pues el modelo aprende a reconocer los elementos en común entre este par de imágenes transformadas. Sin embargo, las transformaciones descritas en la publicación están pensadas para ser utilizadas en fotos, operaciones como alterar colores, aplicar filtros gaussianos o realizar un *zoom* demasiado grande no tienen sentido en un dibujo que solo consiste de lineas en un fondo. Debido a esto se deben proponer nuevas transformaciones para este contexto, teniendo cuidado de que estas no alteren demasiado el dibujo para no perder la semántica. Las transformaciones utilizadas se describen en la Tabla 1.

Transformación	Probabilidad de aplicación	Descripción
Random Line Skip	0.5	Se remueve el 10 % de los trazos del dibujo
Random Rotation	0.5	Se rota el dibujo con un ángulo elegido entre -30° y 30°
Random Horizontal Flip	0.5	Se voltea la imagen horizontalmente
Random Sized Crop	1.0	Se hace un recorte cuadrado de la imagen en un lugar aleatorio, y con un lado de tamaño también aleatorio elegido entre 0.3 y 1 veces el lado de la imagen original

Tabla 1: transformaciones utilizadas para modelo BYOL con dibujos.

3.2.2. BYOL bimodal

Si bien la arquitectura de BYOL es propuesta para aprendizaje auto-supervisado, la idea de tener una red "profesor" de aprendizaje lento, la *target network*, que enseña a otra red "alumno", o la *online network*, es un concepto muy interesante que puede ser extendido a otros problemas. Por esto se propone utilizar una arquitectura BYOL bimodal, es decir, que sea capaz de extraer características de dos modalidades o dominios, en esta caso fotos y dibujos, para enfrentar el problema de recuperación de imágenes basada en dibujos. Para esto se plantea asignar un dominio específico a cada red, de forma de tener una red especializada en dibujos y otra en fotos, aunque la relación de "alumno" y "profesor" se debe determinar de forma experimental, esto se estudia en el Anexo 8.5. Este modelo se entrena con pares de dibujos y fotos del mismo elemento, por lo que no es realmente auto-supervisado pues se necesita generar estos pares, lo que busca es mejorar la generalización evitando utilizar etiquetas y perdidas de clasificación durante el entrenamiento.

Para este modelo se usa el mismo tamaño de imagen, $224 \times 224 \times 3$ y la misma arquitectura que en un BYOL regular. Se utiliza ResNet 50 tanto para la red alumno como la red profesor y *Multi Layer Preceptrons* para los proyectores y el predictor, compuestos de una capa densa o *fully-connected* con salida de tamaño 256 seguido de una capa de normalización y una función de activación *ReLU*. La función de pérdida de un BYOL normal contempla dos distancias con una norma $L2$ al cuadrado, esto debido a que cada transformación pasa dos veces por la red, una por la red alumno y otra por la la red profesor, pero si se quiere que las redes se especialicen en solo dibujos o solo fotos no se debe realizar este cruce de elementos, esta idea se puede ver en la Figura 24. Por esto la función de perdida de este modelo es más simple y consiste de una norma $L2$ al cuadrado en solo una dirección como se ve en la Ecuación 24.

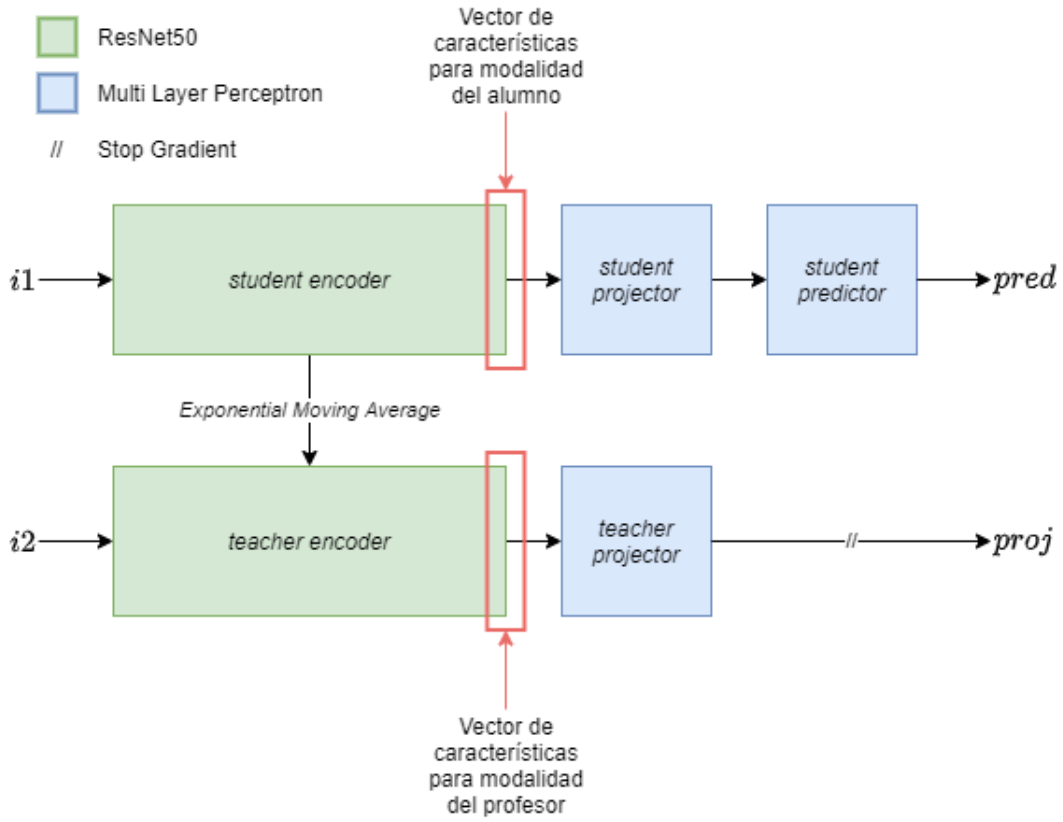


Figura 24: Arquitectura inspirada en BYOL utilizada en los experimentos con dibujos y fotos, la entrada $i1$ corresponde a la entrada de del primer tipo de imagen, en este caso una foto, mientras que $i2$ es el par del otro tipo de imagen, en este caso un dibujo.

$$\text{BYOL Loss} = -\text{SquaredL2Norm}(proj, pred) \tag{24}$$

Como el modelo propuesto es entrenado con pares de dibujos y fotos hay que tener cuidado con las transformaciones realizadas. Se puede considerar que el dibujo ya es una transformación de la foto, una en donde se pierde mucha información y detalles, por lo que aplicar muchas más transformaciones sobre esto puede hacer que las imágenes resultantes no tengan nada en común y se pierda la semántica. En el Anexo 8.6 se realiza un pequeño estudio del efecto de algunas transformaciones en los resultados, en resumen todas las transformaciones evaluadas perjudican a los vectores de características extraídos, por lo que finalmente se decide no utilizar ninguna transformación en este caso.

4. Experimentos y resultados

4.1. Conjuntos de datos utilizados

4.1.1. The Quick, Draw! Dataset

Quick, Draw! es una aplicación creada por Google en donde el usuario juega a dibujar rápidamente distintos objetos cuyos nombres van apareciendo en la pantalla, y luego una red neuronal diseñada por ellos decide si el dibujo es correcto o no. Esta aplicación ha producido una enorme base de datos de dibujos realizados por personas, como los que se pueden ver en la Figura 25, que Google ha dejado para uso libre bajo la licencia Creative Commons con el nombre *The Quick, Draw! Dataset*. Este conjunto de datos presenta las siguientes características:

1. Posee más de 50 millones de dibujos de 345 clases, alrededor de 145 mil dibujos por cada una.
2. Los datos están disponibles como imágenes y también en formato de trazos, reflejando las lineas continuas que realizan los usuarios y el orden en que las dibujaron.
3. Los datos solo están medianamente curados, y por lo tanto, existen algunos dibujos que son *outliers* en cada categoría, por ejemplo un punto o una sola línea.
4. Los dibujos suelen ser simples debido a que el juego impone un límite de tiempo.

Figura 25: Ejemplos de dibujos de *The Quick, Draw! Dataset*.

Este conjunto de datos es bastante grande, trabajar con el total de los datos significa tiempos de entrenamiento demasiado largos que no permitirían entrenar muchos modelos. Por esto se construyen varios subconjuntos de entrenamiento y validación con un número reducido de datos, se estima que alrededor de 1000 ejemplos por clase es una cantidad adecuada para no perder la distribución real de los datos del conjunto. Además se decide dividir el conjunto en dos grupos de 128 clases, la idea es entrenar con solo uno de estos y poder usar otras 128 clases distintas para evaluar la generalización de los modelos, también se evalúa con las mismas 128 clases de entrenamiento para notar la diferencia. Los conjuntos de evaluación tienen 100 ejemplos por clase y tienen todas las etiquetas para poder calcular las métricas necesarias. Los conjuntos definidos se encuentran en la Tabla 2.

Nombre	Descripción	N° de clases	Ejemplos por clase
QD train unlabeled	Dibujos sin etiquetas	128	1000
QD train labeled	Dibujos con todas las etiquetas	128	1000
QD train split 10/90	Dibujos 10 % etiquetados y 90 % no etiquetados	128	1000
QD train split 50/50	Dibujos 50 % etiquetados y 50 % no etiquetados	128	1000
QD eval same	Dibujos de las mismas clases que los conjuntos de entrenamiento con todas las etiquetas	128	100
QD eval other	Dibujos de distintas clases que los conjuntos de entrenamiento con todas las etiquetas	128	100

Tabla 2: Subconjuntos de *The Quick, Draw! Dataset* utilizados.

Estos conjuntos definidos son diseñados para distintos objetivos. El conjunto **QD train unlabeled** que contiene dibujos sin etiquetas está pensado para entrenar los modelos auto-supervisados a estudiar, mientras que el conjunto **QD train labeled** sirve para entrenar modelos supervisados para tener una base de comparación. Los conjuntos **QD train split 10/90** y **QD train split 50/50** son conjuntos del mismo tamaño que los anteriores, pero poseen un porcentaje de los datos con etiquetas y el resto sin estas, estos están diseñados para entrenar los modelos semi-supervisados que se utilizan en este trabajo. Finalmente, se tienen dos conjuntos de evaluación, **QD eval same** que busca evaluar cómo se comportan los modelos al ver las clases con las que fueron entrenados, y **QD eval other** que busca evaluar cómo generalizan hacia otras clases.

4.1.2. Sketchy

El conjunto de datos *Skecthy* [15] es una colección de pares de dibujos y fotos *fine-grained*, es decir, los dibujos están basados en una foto específica y con un alto nivel de detalle como se ve en la Figura 26. Este conjunto está pensado para entrenar modelos en donde se tenga que representar imágenes y fotos en un mismo espacio, como el problema de recuperación de imágenes basada en dibujos. Las características del conjunto son la siguientes:

1. Tiene pares de dibujos y fotos de 125 clases.
2. Posee 12500 fotos de distintos objetos, con 100 elementos por clase.
3. Posee 75471 dibujos de los objetos, con alrededor de 6 dibujos por objeto y 600 dibujos por clase.
4. Las clases de los dibujos son variadas y corresponden a animales, plantas, objetos del hogar, etc.
5. Los dibujos poseen un alto nivel de detalle.

Figura 26: Ejemplos de dibujos y fotos de *Sketchy*.

4.1.3. Flickr15K

Este conjunto de imágenes fue recolectado de sitios web como Flickr, Google y Bing para un estudio sobre recuperación de imágenes basada en dibujos con redes siamesas y *triplet loss* hecho por Bui et al. [16]. Este conjunto de datos es *coarse-grained*, es decir, los dibujos y fotos son de las mismas clases pero no hay un emparejamiento exacto entre estos, como se puede ver en la Figura 27. El conjunto no es demasiado grande y está pensado para ser utilizado como conjunto de evaluación. El conjunto posee las siguientes características:

1. Tiene pares de dibujos y fotos de 33 clases.
2. Posee 14660 fotos de de distintos objetos, con alrededor de 444 elementos por clase.
3. Posee 665 dibujos, con alrededor de 10 dibujos por clase.
4. Las clases de los dibujos son variadas y corresponden a lugares históricos, animales, plantas, etc.
5. Los dibujos poseen un alto nivel de detalle.

Figura 27: Ejemplos de dibujos y fotos de *Flickr15K*.

4.1.4. Conjunto de datos de *eCommerce*

Este conjunto de datos es propuesto por Pablo Torres y José Saavedra en su trabajo sobre representaciones compactas para el problema de recuperación de imágenes basada en dibujos [7], con el fin de representar mejor el caso de uso en *e-commerce*. Este conjunto de datos posee pares de fotos de productos que se pueden encontrar en el *e-commerce* con dibujos de las mismas clases, como los de la Figura 28, no es demasiado grande y está pensado para ser usado como conjunto de evaluación. Esta colección de datos también es *coarse-grained*, y por lo tanto no hay un emparejamiento exacto entre dibujos y fotos. El conjunto tiene las siguientes características:

1. Tiene pares de dibujos y fotos de 133 clases.
2. Posee 10600 fotos de distintos objetos, con alrededor de 80 elementos por clase.
3. Posee 665 dibujos, con alrededor de 5 dibujos por clase.
4. Las clases corresponden a productos que se pueden encontrar en una tienda de *e-commerce*
5. Los dibujos poseen un alto nivel de detalle.

Figura 28: Ejemplos de dibujos y fotos de *eCommerce*.

4.2. Métricas utilizadas

4.2.1. Accuracy de un clasificador *K-Nearest Neighbors*

Un clasificador KNN asigna a un elemento la clases más frecuente entre sus vecinos, es una buena forma de evaluar que los elementos de los espacios latentes generados estén agrupados por clase. En general este clasificador se utiliza con valores de K pequeños, se estima que para este experimento un valor de $K = 5$ es suficiente para poder comparar modelos. El valor utilizado finalmente, es el *accuracy* de este clasificador, es decir, el porcentaje de elementos del conjunto de evaluación cuya predicción es correcta, como se ve en la Ecuación 25.

$$\text{accuracy} = \frac{\text{número de predicciones correctas}}{\text{número de predicciones totales}} \tag{25}$$

4.2.2. Mean Average Precision

Una métrica que evalúa de forma más directa las consultas realizadas es **mAP**, en donde efectivamente se realizan varias consultas y se entrega un promedio de todas estas. En el caso de los modelos que solo utilizan dibujos se tiene solo un conjunto de evaluación, de esta forma **mAP** se obtiene utilizando cada dibujo del conjunto para realizar una consulta y se descarta este mismo de los resultados. En este contexto se utiliza **mAP@5** que solo considera los primeros 5 elementos de la clase correcta, esto debido a que es menos costoso computacionalmente y permite evaluar con conjuntos más grandes y realizar más experimentos. En el caso en que se hacen consultas de dibujos para recuperar fotos se utilizan un dos conjuntos de evaluación, uno con dibujos y otro con fotos, y se usa **mAP** de forma normal, considerando todos los elementos recuperados, esto para poder comparar resultados con otros trabajos relacionados.

4.2.3. t-SNE

Para poder entender los espacios de alta dimensionalidad generados por los modelos, se utiliza el método de reducción de dimensionalidad **t-SNE** [17]. Este método está pensado para realizar visualizaciones de grandes conjuntos de datos de alta dimensionalidad, el método se preocupa de mantener a los vecinos, o elementos cercanos, durante la transformación y además evita producir acumulaciones de muchos datos. En este caso se utiliza para reducir los vectores de características obtenidos a dos dimensiones y se muestra la distribución de las clases en los espacios latentes de los modelos.

4.3. Ajustes Experimentales

4.3.1. Modelos basados en VAE

1. Se utiliza optimizador Adam con un learning rate de $3 \cdot 10^{-4}$.
2. Se entrena hasta 30 epochs, eligiendo el mejor modelo al final de un epoch según las métricas de evaluación.
3. Se utilizan los dibujos con un tamaño de 256×256, escalados entre 0 y 1.
4. Se utiliza un espacio latente de 32 dimensiones, justificación en Anexo 8.1.
5. Se utiliza un peso β igual a 0,1, justificación en Anexo 8.2.

4.3.2. Modelos basados en BYOL

1. Se utiliza optimizador Adam con un learning rate de $3 \cdot 10^{-4}$.
2. Se entrena hasta 100 epochs, eligiendo el mejor modelo al final de un epoch según las métricas de evaluación.
3. Se utilizan los dibujos con un tamaño de $224 \times 224 \times 3$, con su valores originales entre 0 y 255.
4. Se utiliza *cosine decay* para el parámetro τ del *exponential moving average.*

4.4. Resultados de modelos basados en VAE

4.4.1. VAE

En estos experimentos se entrena un modelo VAE con el conjunto **QD train unlabeled** con el fin evaluar las características visuales extraídas por el modelo. Se quiere evaluar la viabilidad de estos para realizar búsquedas de dibujos para lo cual se comparan los resultados de las métricas obtenidas con las de un modelo ResNet 50 entrenado de forma supervisada con el conjunto **QD train labeled**. Además se realizan visualizaciones del espacio latente utilizando el método t-SNE para reducir la dimensionalidad del espacio latente a dos dimensiones,

para entender cómo se distribuyen las clases en el espacio. Luego se realizan consultas reales de dibujos para poder evaluar de forma cualitativa si los dibujos recuperados son relevantes.

Al comparar los vectores de características producidos por este modelo con los de un ResNet 50 entrenado de forma supervisada, se puede notar que los resultados están bastante lejos de competir contra un modelo supervisado. En el caso de evaluar con las mismas clases usadas en el entrenamiento, como se ve en la Tabla 3 , se puede ver que VAE obtiene valores en las métricas que corresponden a la mitad de los valores obtenidos por ResNet 50. Al evaluar con clases distintas a las del entrenamiento, Tabla 4, se ve que aunque las métricas de ResNet 50 bajan, siguen siendo considerablemente mejores. Algo interesante de VAE es que logra resultados parecidos en ambos conjuntos de datos, esto se puede atribuir a que el modelo no utiliza ninguna etiqueta por lo extrae características que no dependen de estas.

	knn accuracy k=5	**mAP@5**
VAE ResNet	0,338	0,307
Supervised ResNet	0,687	0,655

Tabla 3: Resultados de evaluación en el conjunto **QD eval same** vs. modelo supervisado.

	knn accuracy k=5	**mAP@5**
VAE ResNet	0,330	0,310
Supervised ResNet	0,575	0,528

Tabla 4: Resultados de evaluación en el conjunto **QD eval other** vs. modelo supervisado.

Al visualizar la distribución de clases en el espacio latente, en la Figura 29, se puede ver que estas ocupan casi todo el espacio, esto sucede tanto con las clases utilizadas en el entrenamiento como en otras. Si bien es posible notar como se agrupan los elementos de las mismas clases, viendo los clusters de colores que se generan, también se ve que existen grandes intersecciones entre clases, además de *outliers* que se encuentran lejos de su cluster. Estas propiedades parecen explicar los bajos resultados obtenidos en las métricas obtenidas en las Tablas 3 y 4.

(a) Clases de **QD eval same** (b) Clases de **QD eval other**

Figura 29: Visualización en dos dimensiones de 8 clases en el espacio latente de un VAE.

Al observar ejemplos de búsquedas de dibujos, en las Figuras 30 y 31, se puede ver que en algunas clases el modelo si es capaz de encontrar otros dibujos similares del mismo contenido. Sin embargo, también se ve como se obtienen resultados erróneos con dibujos que no son necesariamente complejos, por ejemplo en el caso de los pantalones o la flor. En estos casos los resultados si tienen algo de similitud con la consulta, pero pareciera que el modelo VAE considera principalmente los contornos de los dibujos en las características extraídas, sin prestar mucha atención a los detalles.

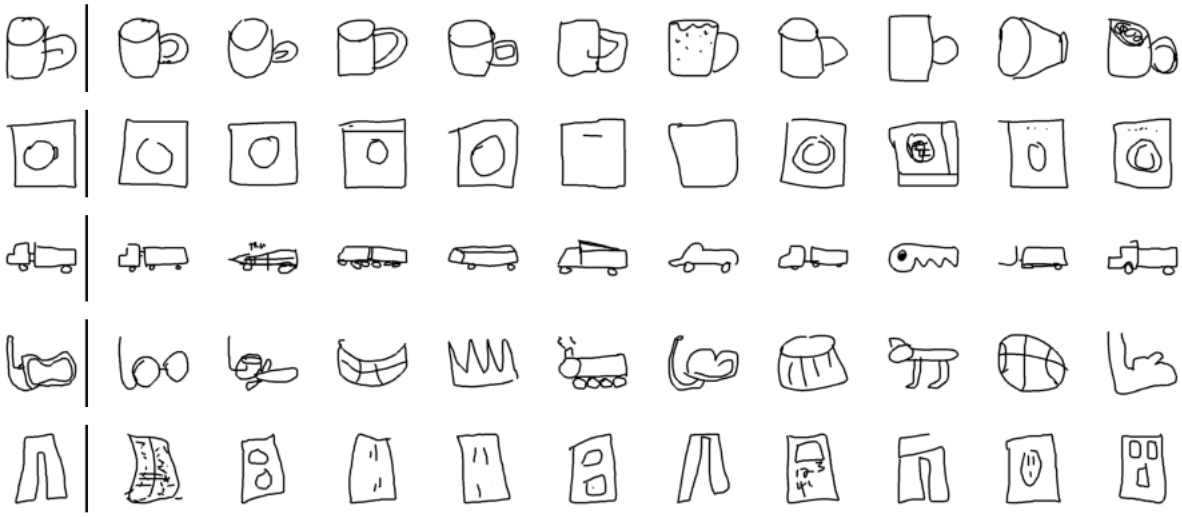


Figura 30: Ejemplos de búsquedas de dibujos con VAE en el conjunto **QD eval same**.

Figura 31: Ejemplos de búsquedas de dibujos con VAE en el conjunto **QD eval other**.

4.4.2. Modelo M2

En estos experimentos se entrena un modelo M2 con dos conjuntos: **QD train split 10/90** y **QD train split 50/50** con el fin evaluar si agregar un porcentaje de datos no etiquetados mejora los resultados producidos por un modelo AlexNet supervisado. Para ver si efectivamente existe se entrenan modelos AlexNet que utilizan solo los datos con etiquetas de los conjuntos utilizados, el resultado esperado es que los modelos semi-supervisados mejoren las métricas sobre esta cota inferior. También se entrena una cota superior entrenada como si todo el conjunto estuviera etiquetado, para ver que tan lejos se está del mejor caso. Luego se realizan visualizaciones del espacio latente utilizando el método t-SNE, reduciendo la dimensionalidad del espacio latente a dos dimensiones, para ver cómo se distribuyen las clases en el espacio. También se realizan consultas reales de dibujos para poder tener una evaluación cualitativa de los resultados.

En la Tabla 5, se ven los resultados de entrenar un modelo M2 entrenado con un 10 % de datos etiquetados y 90 % de datos no etiquetados para realizar búsquedas, evaluando con las mismas clases utilizadas en el entrenamiento. Se puede ver que el modelo M2 presenta mejores resultados que un modelo entrenado de forma supervisada con el 10 % etiquetado, y peores que el caso de tener todos los datos etiquetados, lo cual es el efecto esperado. Esto cambia si evaluamos en clases no contenidas en el conjunto de entrenamiento, como se muestra en la Tabla 6, donde las métricas de los modelos M2 empeoran significativamente, pasan a ser peores que el modelo supervisado entrenado con el 10 % etiquetado.

	knn accuracy k=5	**mAP@5**
AlexNet 10 % etiquetado	0,520	0,482
M2 10 % etiquetado 90 % no etiquetado	0,528	0,492
AlexNet 100 % etiquetado	0,620	0,585

Tabla 5: Modelo M2 entrenado con **QD train split 10/90** evaluado en el conjunto **QD eval same** vs. modelos supervisados.

	knn accuracy k=5	**mAP@5**
AlexNet 10 % etiquetado	0,496	0,452
M2 10 % etiquetado 90 % no etiquetado	0,291	0,267
AlexNet 100 % etiquetado	0,526	0,485

Tabla 6: Modelo M2 entrenado con **QD train split 10/90** evaluado en el conjunto **QD eval other** vs. modelos supervisados.

Si se aumenta la proporción de datos etiquetados al 50 %, las métricas en el conjunto con las mismas clases usadas en el entrenamiento suben de forma considerable, esto se puede ver en la Tabla 7, sobrepasando incluso al modelo entrenado con todos los datos de forma supervisada. Esto es interesante pues puede tener un caso de uso real si solo se busca trabajar con un conjunto fijo de clases. Al parecer las propiedades de VAE que hereda el modelo M2 permiten organizar el espacio de forma que mejora los resultados de búsqueda de dibujos. Por otro lado, las métricas en el conjunto con otras clases empeoran nuevamente, como se ve en la Tabla 8, el modelo M2 no parece ser bueno para extraer características generalizadas.

	knn accuracy k=5	**mAP@5**
AlexNet 50 % etiquetado	0,595	0,559
M2 50 % etiquetado 50 % no etiquetado	0,663	0,624
AlexNet 100 % etiquetado	0,620	0,585

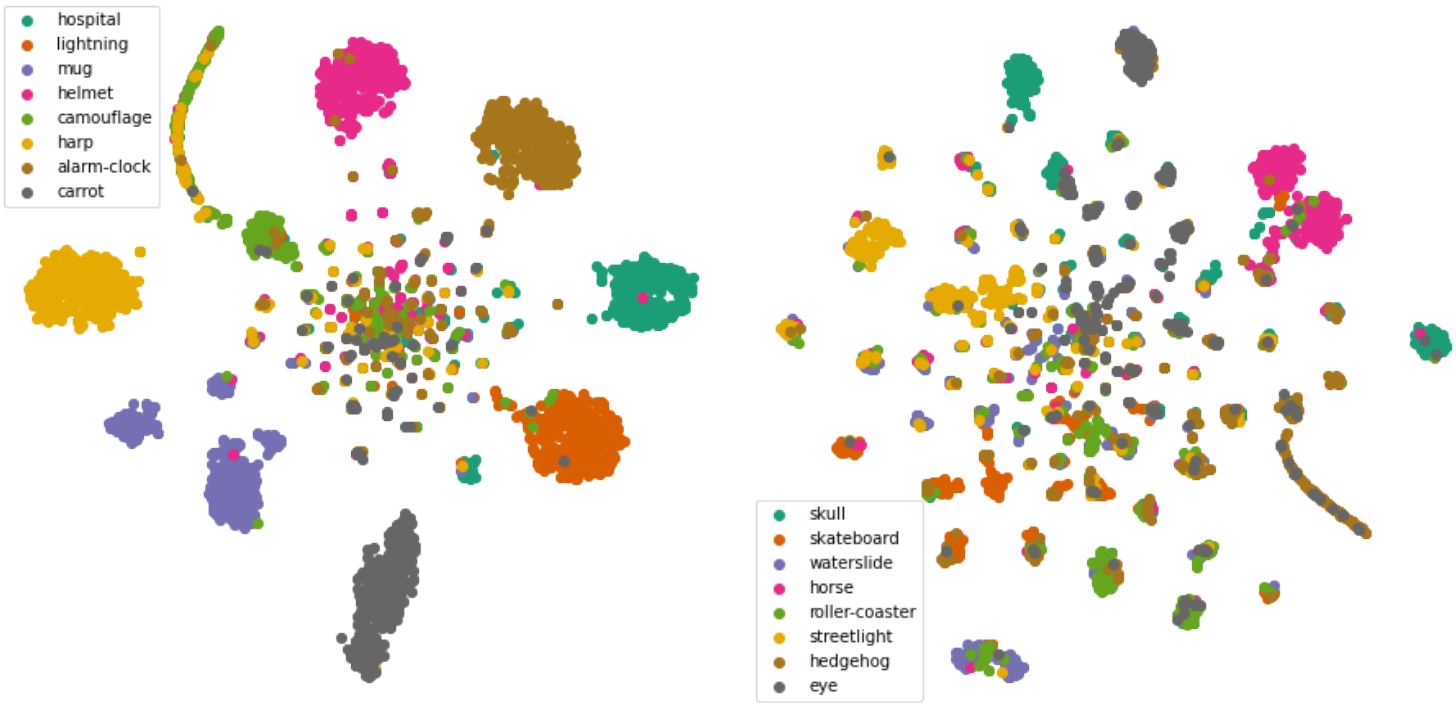
Tabla 7: Modelo M2 entrenado con **QD train split 50/50** evaluado en el conjunto **QD eval same** vs. modelos supervisados.

	knn accuracy k=5	**mAP@5**
AlexNet 50 % etiquetado	0,530	0,484
M2 50 % etiquetado 50 % no etiquetado	0,318	0,298
AlexNet 100 % etiquetado	0,526	0,485

Tabla 8: Modelo M2 entrenado con **QD train split 50/50** evaluado en el conjunto **QD eval other** vs. modelos supervisados.

Al visualizar en dos dimensiones el espacio latente, Figura 32, se puede ver una clara diferencia entre las clases que pertenecen al conjunto de entrenamiento en comparación al resto. En el primer caso se forman grupos bien definidos que corresponden a las clases utilizadas,

aunque con un grupo de elementos en el centro en donde se encuentran varios elementos de distintas clases. Al observar otras clases, se ven pequeños grupos en todo el espacio en donde algunos efectivamente contienen solo una clase, pero varios presentan intersecciones entre varias más. Estos resultados coinciden con los vistos en las métricas de las tablas anteriores.



(a) Clases de **QD eval same** (b) Clases de **QD eval other**

Figura 32: Visualización en dos dimensiones de 8 clases en el espacio latente de un M2.

Al ver los ejemplos de búsqueda con las mismas clases de entrenamiento, en la Figura 33, se puede ver que los resultados son mejores que los obtenidos con un VAE regular, incluso con los dibujos más complejos los dibujos recuperados son relevantes. Por otro lado, al realizar búsquedas con otras clases los resultados empeoran bastante, como se ve en la Figura 34. Muchos de los dibujos recuperados en este caso no parecen tener nada en común con la consulta, una posible explicación es que las características obtenidas solo tengan sentido en el contexto de las clases usadas para el entrenamiento. Nuevamente los resultados cualitativos corroboran las métricas obtenidas del modelo M2.

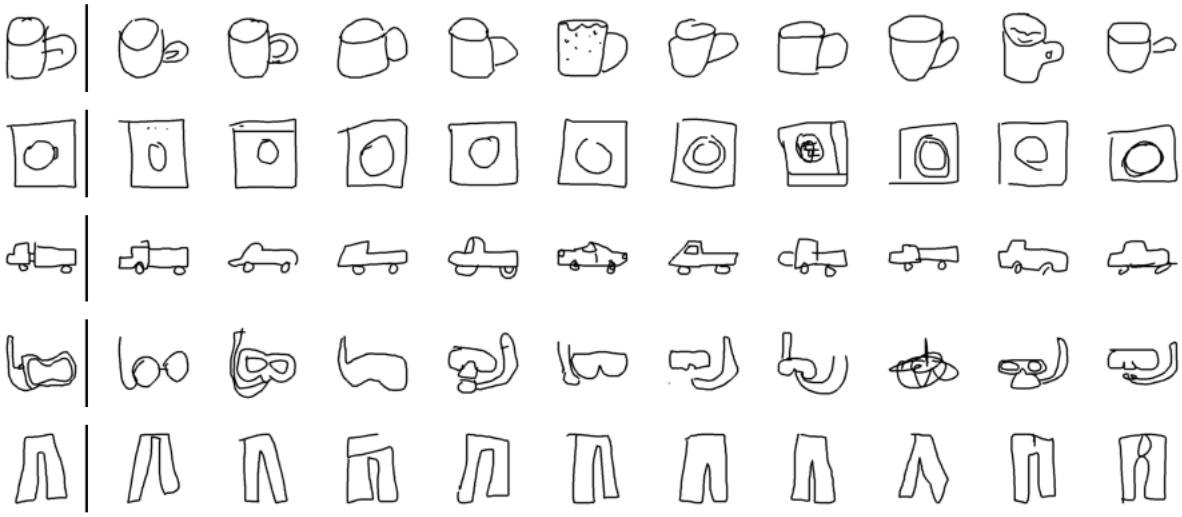


Figura 33: Ejemplos de búsquedas de dibujos con M2 en el conjunto **QD eval same**.

Figura 34: Ejemplos de búsquedas de dibujos con M2 en el conjunto **QD eval other**.

4.4.3. VAE semi-supervisado

En estos experimentos se entrena un modelo VAE semi-supervisado con dos conjuntos: **QD train split 10/90** y **QD train split 50/50** para evaluar si agregar un porcentaje de datos no etiquetados mejora los resultados producidos por un modelo AlexNet supervisado. Al igual que con el modelo M2, para ver si existe una mejora se entrenan modelos AlexNet que utilizan solo los datos con etiquetas de los conjuntos mencionados, se espera que modelos semi-supervisados logren mejorar las métricas sobre esta cota inferior. También se entrena un modelo AlexNet con todos los datos etiquetados como una cota superior. Luego se realizan visualizaciones del espacio latente utilizando el método t-SNE, reduciendo la dimensionalidad del espacio latente a dos dimensiones y se realizan consultas reales de dibujos para poder tener una evaluación cualitativa de los resultados.

Al evaluar el modelo VAE semi-supervisado entrenado con el conjunto *QD train split 10/90*, utilizando las mismas clases utilizadas en el entrenamiento, como se ve en la Tabla 9, se puede ver que los resultados son peores que incluso la cota inferior. En el caso de otras clases, Tabla 10, el modelo presenta los mismos problemas para generalizar a otras clases que el modelo M2, aunque la diferencia según el grupo de clases utilizadas parece ser menos pronunciada en este caso. De esta forma se puede apreciar que el modelo VAE supervisado no logra mejorar los resultados agregando las etiquetas del 10 % de los dibujos, cosa que el modelo M2 si fue capaz de realizar.

	knn accuracy k=5	**mAP@5**
AlexNet 10 % etiquetado	0,520	0,482
SSVAE 10 % etiquetado 90 % no etiquetado	0,490	0,449
AlexNet 100 % etiquetado	0,620	0,585

Tabla 9: VAE semi-supervisado entrenado con **QD train split 10/90** evaluado en el conjunto **QD eval same** vs. modelos supervisados.

	knn accuracy k=5	**mAP@5**
AlexNet 10 % etiquetado	0,496	0,452
SSVAE 10 % etiquetado 90 % no etiquetado	0,403	0,358
AlexNet 100 % etiquetado	0,526	0,485

Tabla 10: VAE semi-supervisado entrenado con **QD train split 10/90** evaluado en el conjunto **QD eval other** vs. modelos supervisados.

Al subir la razón de datos etiquetados al 50 %, con el conjunto *QD train split 50/50*, las métricas pasan a ser parecidas a las del modelo M2 en el caso de las mismas clases usadas para el entrenamiento. Como se muestra en la Tabla 11, también se logra superar al mejor caso en donde se tienen todas las etiquetas. Sin embargo, nuevamente al probar con otras clases, Tabla 12, el modelo VAE semi-supervisado no logra obtener una buena generalización, teniendo peores resultados que la cota inferior entrenada con solo con el 50 % de los datos etiquetados. Ninguno de los dos modelos semi-supervisados logra generalizar hacia otras clases.

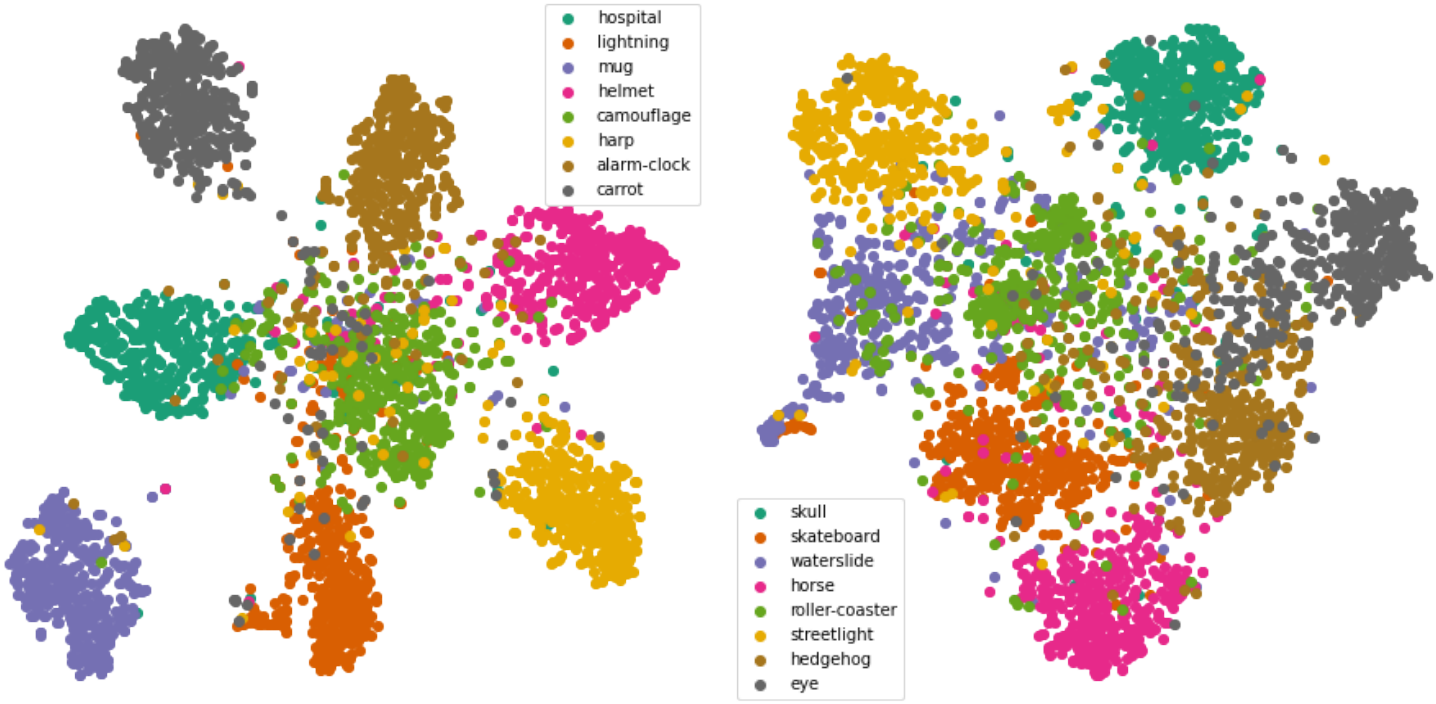
	knn accuracy k=5	**mAP@5**
AlexNet 50 % etiquetado	0,595	0,559
SSVAE 50 % etiquetado 50 % no etiquetado	0,672	0,648
AlexNet 100 % etiquetado	0,620	0,585

Tabla 11: VAE semi-supervisado entrenado con **QD train split 50/50** evaluado en el conjunto **QD eval same** vs. modelos supervisados.

	knn accuracy k=5	**mAP@5**
AlexNet 50 % etiquetado	0,530	0,484
SSVAE 50 % etiquetado 50 % no etiquetado	0,422	0,390
AlexNet 100 % etiquetado	0,526	0,485

Tabla 12: VAE semi-supervisado entrenado con **QD train split 50/50** evaluado en el conjunto **QD eval other** vs. modelos supervisados.

En la visualización en dos dimensiones del espacio latente generado, en la Figura 35, se puede ver que las clases utilizadas en el entrenamiento se agrupan en conjuntos bien definidos, con algunos elementos fuera de lugar en el centro. En el caso de otras clases, el espacio latente pasa a ser similar al de un VAE regular, en donde se ocupa una mayor parte del espacio, y aunque aún hay agrupaciones de elementos de las mismas clases, hay una mayor intersección entre ellas. Se ve que el efecto de agregar etiquetas es "apretar" a los elementos de las clases correspondientes en el espacio latente, pero sin afectar al resto de las clases.



(a) Clases de **QD eval same** (b) Clases de **QD eval other**

Figura 35: Visualización en dos dimensiones de 8 clases en el espacio latente de un VAE semi-supervisado.

En los ejemplos de consultas con dibujos de las mismas clases usadas en el entrenamiento, en la Figura 36, se puede apreciar que en general se obtienen buenos resultados a excepción de la máscara de buceo o snorkel, que es un dibujo más complejo. En el caso de otras clases, se puede ver una pérdida en la calidad de los resultados, aunque no tan pronunciada como en el caso del modelos M2, estos resultados son comparables con los obtenidos con VAE regular, sino un poco mejor.

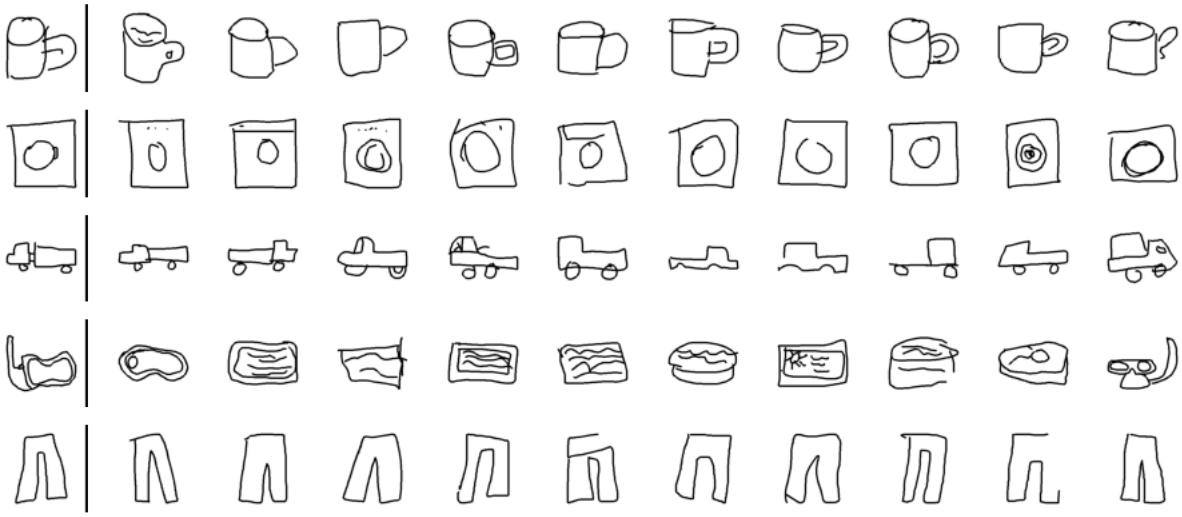


Figura 36: Ejemplos de búsquedas de dibujos con VAE semi-supervisado en el conjunto **QD eval same**.

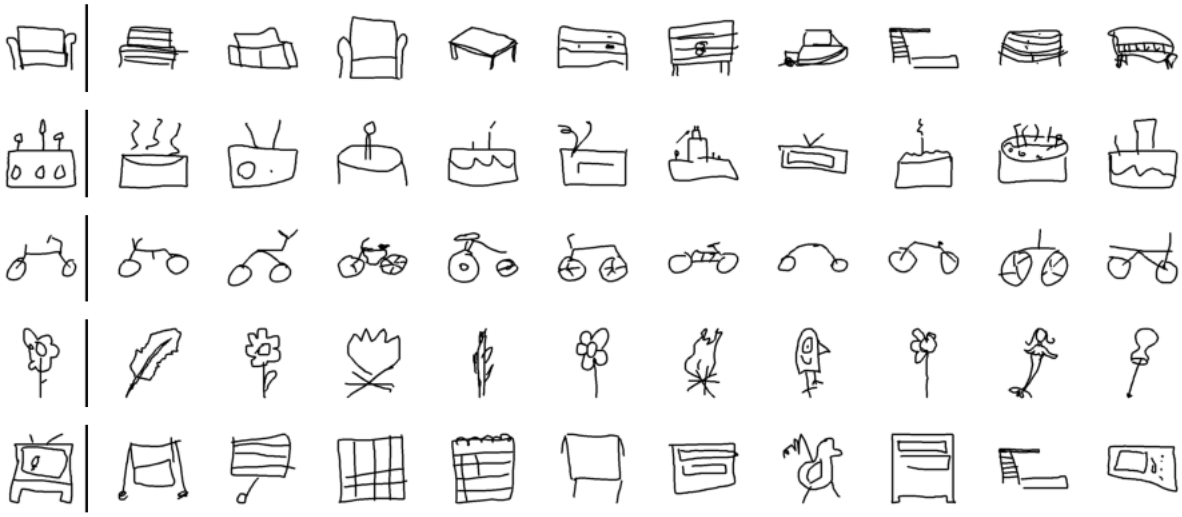


Figura 37: Ejemplos de búsquedas de dibujos con VAE semi-supervisado en el conjunto **QD eval other**.

4.5. Resultados modelos basados en BYOL

4.5.1. BYOL

Debido a los resultados obtenidos con los modelos generativos de la familia VAE, se decide probar como se comporta el modelo discrimantivo BYOL para extraer características visuales de dibujos. En estos experimentos se entrena un modelo BYOL con el conjunto **QD train unlabeled** y se procede a evaluar las características visuales extraídas por el modelo. Se evalúan los valores de las métricas obtenidas con con las de un modelo ResNet 50 entrenado

de forma supervisada con el conjunto **QD train labeled**. A continuación, se realizan visualizaciones del espacio latente utilizando el método t-SNE reduciendo la dimensionalidad del espacio latente a dos dimensiones, y finalmente, se realizan consultas reales de dibujos para poder evaluar que tan relevantes son los dibujos recuperados.

Al comparar la capacidad de los vectores de características obtenidos para realizar búsquedas de dibujos con un modelo ResNet supervisado, se puede ver que BYOL logra resultados competitivos en las métricas. Como se ve en la Tabla 13, BYOL se encuentra a solo un 5 %, en ambas métricas de alcanzar a ResNet cuando se evalúa en las mismas clases usadas en el entrenamiento, además logrando resultados bastante superiores a los de VAE, que también es un modelo auto-supervisado. Al evaluar con otras clases los resultados son aún mejores, como se ve en la Tabla 14, BYOL logra generalizar mejor que el modelo supervisado, pues los resultados de las métricas de BYOL se mantienen mientras que las de ResNet bajan. Las tablas 13 y 14 también tienen los resultados obtenidos previamente con el modelo VAE, en este aspecto es interesante ver como las métricas de ambos métodos auto-supervisados no se ven afectadas cuando se cambian las clases con las que se esta trabajando.

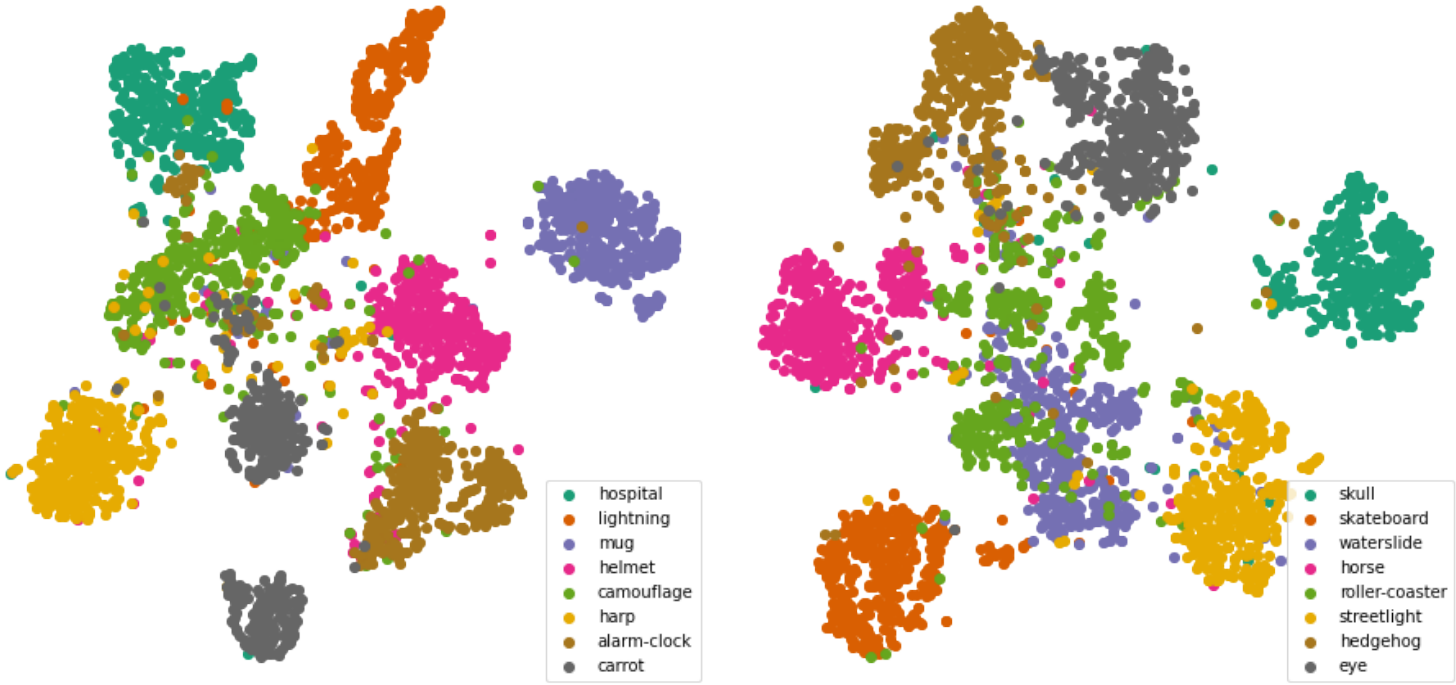
	knn accuracy k=5	**mAP@5**
VAE	0,338	0,307
BYOL	0,634	0,597
ResNet50	0,687	0,655

Tabla 13: BYOL evaluado en el conjunto **QD eval same** vs. modelos supervisados.

	knn accuracy k=5	**mAP@5**
VAE	0,330	0,310
BYOL	0,627	0,590
ResNet50	0,575	0,528

Tabla 14: BYOL evaluado en el conjunto **QD eval other** vs. modelos supervisados.

En la Figura 38, al observar la visualización en dimensiones del espacio latente generado, se ve, de forma similar al espacio latente de un VAE, que no existe una gran diferencia al comparar la distribución de las clases del entrenamiento con otras fuera del conjunto. Sin embargo, también es posible observar que estas distribuciones no utilizan todo el espacio como sucede con un VAE, se forman grupos bien definidos y que se distancian unos de otros, lo que coincide con los buenos resultados obtenidos con BYOL.



(a) Clases de **QD eval same** (b) Clases de **QD eval other**

Figura 38: Visualización en dos dimensiones de 8 clases en el espacio latente de un BYOL.

Si se observan los resultados de consultas de dibujos con BYOL, en las Figuras 39 y 40, se puede ver se obtienen buenos resultados tanto para las mismas clases del entrenamiento como en otras. En cada una de las consultas la mayoría de los resultados obtenidos son relevantes. Una posible explicación de estos buenos resultados es que como BYOL utiliza constantemente transformaciones durante el entrenamiento, entonces aprende a extraer características sobre detalles más específicos que solo la forma exterior, que es una de las debilidades del modelo VAE.

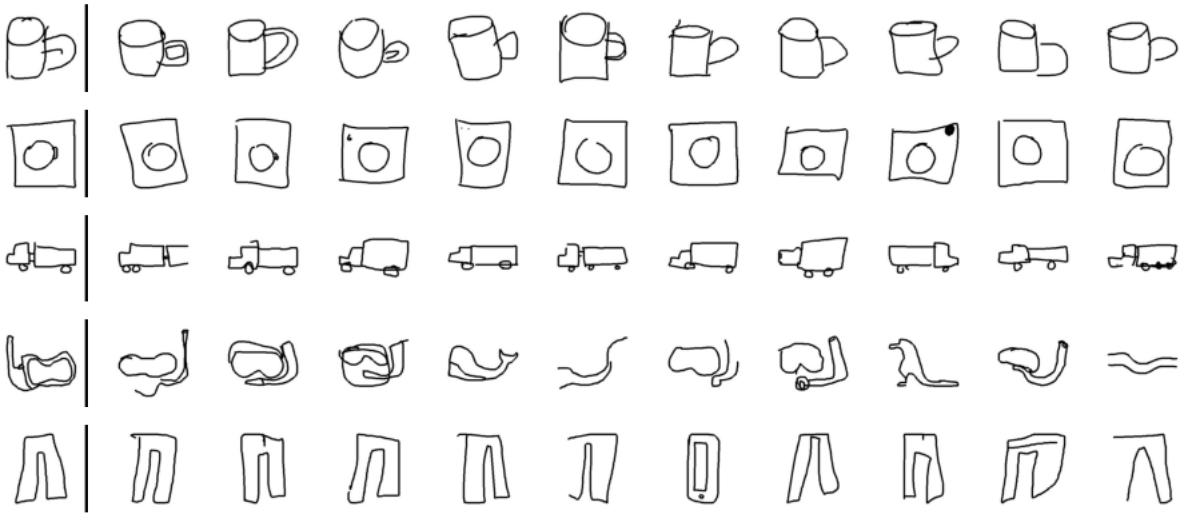


Figura 39: Ejemplos de búsquedas de dibujos con BYOL en el conjunto **QD eval same**.

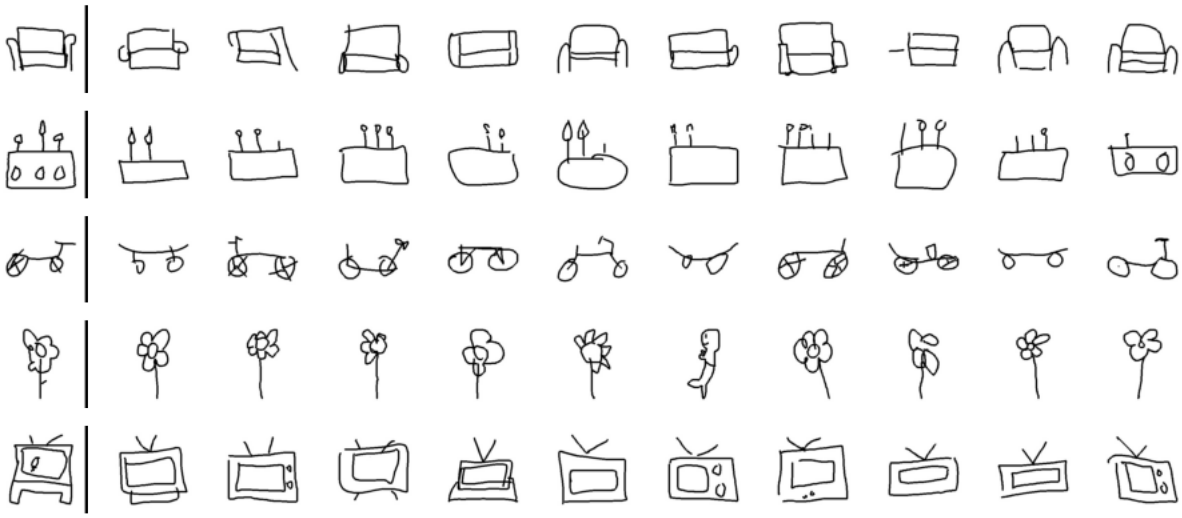


Figura 40: Ejemplos de búsquedas de dibujos con BYOL en el conjunto **QD eval other**.

4.5.2. BYOL bimodal

En este experimento se busca evaluar un modelo BYOL extendido para trabajar con fotos y dibujos en el contexto de recuperación de imágenes basada en dibujos. Este se entrena sin utilizar ninguna etiqueta, y por lo tanto se espera que sea capaz de generalizar de mejor manera a otros conjuntos de datos. Para poder evaluar la efectividad de los vectores de características obtenidos por el modelo BYOL bimodal, se usa como base de comparación el modelo de red siamesa entrenado con múltiples etapas en el trabajo de Pablo Torres y José Saavedra [7]. En la Tabla 15 se puede ver como BYOL bimodal muestra resultados cercanos a los obtenidos con la red siamesa al evaluar con el conjunto *Flickr15K*. Ambos modelos fueron entrenados con *Sketchy*. Si bien el resultado de BYOL bimodal es menor, esto se logra sin utilizar las etiquetas junto a una pérdida de clasificación, como se hace en la red siamesa.

	mAP
BYOL bimodal	0,351
Red siamesa entrenada con múltiples etapas	0,421

Tabla 15: BYOL bimodal evaluado en el conjunto *Flickr15K* vs. modelo supervisado.

Por otro lado, al evaluar con el conjunto *eCommerce*, como se muestra en la Tabla 16, se ve que BYOL bimodal pasa a tener mejores resultados. Esto puede suceder debido a que el conjunto de *eCommerce* es bastante distinto al conjunto *Sketchy* utilizado para el entrenamiento, de forma que la red siamesa se "especializa" en las clases de *Sketchy*, mientras que el modelo BYOL bimodal logra generalizar mejor por no utilizar etiquetas.

	mAP
BYOL bimodal	0,178
Red siamesa entrenada con multiples etapas	0,145

Tabla 16: BYOL bimodal evaluado en el conjunto *eCommerce* vs. modelo supervisado.

Al observar los ejemplos de consultas realizados en el conjunto *Flickr15K*, Tabla 41, se puede ver que el modelo es capaz de detectar formas simples sin problemas, pero en los casos complejos, como la tercera consulta que contiene un dibujo del coliseo romano, recupera resultados incorrectos. Por otro lado, en las consultas en el conjunto de *eCommerce*, los resultados no muestran ser muy buenos, aunque el primer objeto recuperado si tiene relación con el dibujo, también existen muchos objetos recuperados que no tienen mucha relacion. Si bien los resultados obtenidos en las métricas parecen ser buenas en comparación al método supervisado, si se realiza una evaluación cualitativa de las búsquedas todavía se ve que queda espacio para mejorar.

Figura 41: Ejemplos de búsquedas de dibujos con BYOL bimodal en el conjunto *Flickr15K*.

Figura 42: Ejemplos de búsquedas de dibujos con BYOL en el conjunto **eCommerce**.

4.6. Resumen de resultados en el contexto de dibujos

Debido a que en este trabajo se evalúa una gran cantidad de modelos en el contexto de extracción de características de dibujos, a continuación se muestran los resultados de todos los modelos condensados en dos tablas, en la Tabla 17 se tienen los resultados al evaluar con las mismas clases de entrenamiento, mientras que en la Tabla 18 se evalúa con clases distintas a las del entrenamiento.

	knn accuracy k=5	**mAP@5**	**Tipo de entrenamiento**
VAE	0,338	0,307	auto-supervisado
M2 10 % etiquetado	0,528	0,492	semi-supervisado
M2 50 % etiquetado	0,663	0,624	semi-supervisado
SSVAE 10 % etiquetado	0,490	0,449	semi-supervisado
SSVAE 50 % etiquetado	0,672	0,648	semi-supervisado
BYOL	0,634	0,597	auto-supervisado
ResNet	**0,687**	**0,655**	**supervisado**

Tabla 17: Resumen de resultados de modelos para extracción de características de dibujos, evaluados con el conjunto **QD eval same** con las mismas clases de entrenamiento.

	knn accuracy k=5	**mAP@5**	**Tipo de entrenamiento**
VAE	0,330	0,310	auto-supervisado
M2 10 % etiquetado	0,291	0,267	semi-supervisado
M2 50 % etiquetado	0,318	0,298	semi-supervisado
SSVAE 10 % etiquetado	0,403	0,358	semi-supervisado
SSVAE 50 % etiquetado	0,422	0,390	semi-supervisado
BYOL	**0,627**	**0,590**	**auto-supervisado**
ResNet	0,575	0,528	supervisado

Tabla 18: Resumen de resultados de modelos para extracción de características de dibujos, evaluados con el conjunto **QD eval other** con clases distintas a las de entrenamiento.

5. Conclusiones

En este trabajo se implementaron y evaluaron distintos modelos, tanto auto-supervisados como semi-supervisados, para la extracción de características visuales. Los modelos estudiados se dividen en dos grupos: los modelos basados en *Variational Autoencoders* o VAE y los modelos basados en *Bootstrap your own latent* o BYOL. Dentro de la primera familia de modelos, primero se implementó VAE y se entrenó con dibujos sin utilizar ninguna etiqueta. Este modelo no logra resultados competitivos con métodos supervisados, pero posee la característica de que al no utilizar etiquetas, obtiene características que funcionan de la misma manera al cambiar las clases con las que se evalúa. Además VAE parece enfocarse solo en la forma general de un dibujo, sin considerar los detalles más específicos.

Luego se buscó estudiar la posibilidad de extender el modelo VAE para poder incluir un porcentaje de dibujos con etiquetas al entrenamiento, de esta forma se implementaron los modelos M2 y VAE semi-supervisado, que fueron entrenados con un conjunto de dibujos del mismo tamaño que VAE, pero con un 10 % y un 50 % de los datos con etiquetas respectivamente. Estos modelos logran resultados competitivos, incluso superiores en algunos casos, a un modelo supervisado en el caso de evaluar con las clases utilizadas en el entrenamiento. Esto puede ser útil si se necesita un modelo que solo trabaje con un conjunto fijo de clases. Sin embargo, al evaluar con otras clases ambos modelos obtienen métricas mucho peores, parecidos a los de un VAE regular. Dentro de estos dos modelos se destaca el modelo M2, que es capaz de mejorar resultados usando solo un 10 % de datos etiquetados, el modelo VAE semi-supervisado solo logra buenos resultados al utilizar un 50 % de los datos etiquetados.

La familia de modelos basados en VAE no logra buenos resultados en otras clases no contempladas en el entrenamiento, el exigir a un modelo que sea capaz de codificar toda la información necesaria para poder reconstruirlo, parece afectar de forma negativa a los vectores de características obtenidos. En general VAE parece tomar en cuenta sólo los contornos de los dibujos, sin prestar atención a los detalles. Por esto se procedió a implementar el modelo BYOL, un modelo discriminativo que utiliza transformaciones durante el entrenamiento, y se procedió a entrenarlo con dibujos sin utilizar etiquetas. BYOL logra resultados competitivos con los modelos supervisados, al evaluar tanto con las mismas clases que utilizó en el entrenamiento como con otras. Al igual que un modelo VAE los resultados se mantienen similares al cambiar de clases, pues no utiliza etiquetas durante el entrenamiento, pero BYOL sí considera los detalles del dibujo obteniendo resultados muy superiores.

Finalmente, se implementó un modelo basado en BYOL que es capaz de trabajar tanto con dibujos como con fotos, para enfrentar el problema de recuperación de imágenes basada en dibujos. Este modelo se entrenó con pares de fotos y dibujos realizados específicamente para cada ejemplo. Luego, BYOL bimodal logra resultados competitivos con los modelos que usan etiquetas, logrando resultados un poco más bajos en el conjunto *Flickr15K* y un poco más altos en el conjunto de *eCommerce*. Considerando que el conjunto de *eCommerce* presenta características distintas a las del conjunto de entrenamiento, los resultados indican que BYOL bimodal, al igual que los modelos auto-supervisados, presenta mejores capacidades de generalización, lo que puede tener un impacto en la industria, permitiendo mejorar los resultados obtenidos al entrenar con los conjuntos de datos de dibujos y fotos existentes.

6. Trabajo futuro

Si bien el modelo BYOL bimodal logra deshacerse de las etiquetas para el entrenamiento, el modelo aún necesita de pseudo etiquetas al utilizar pares de fotos con dibujos hechos específicamente para cada foto. Sería interesante poder estudiar una forma de poder generar un dibujo para cada foto de forma automática, a través de algún algoritmo, de forma que el modelo pase a ser completamente auto-supervisado. Esto permitiría poder utilizar conjuntos de datos con solo fotos, sin etiquetas, para el entrenamiento, que son mucho más comunes que los conjuntos de pares que se necesitan hoy en día. Esto también significa poder encontrar conjuntos de entrenamiento que se ajusten mejor al problema que se busque resolver. Una posible opción es utilizar algoritmos de detección de bordes para generar una pseudo dibujo, generando así el par que necesita BYOL bimodal, esta idea se puede ver en la Figura 43.

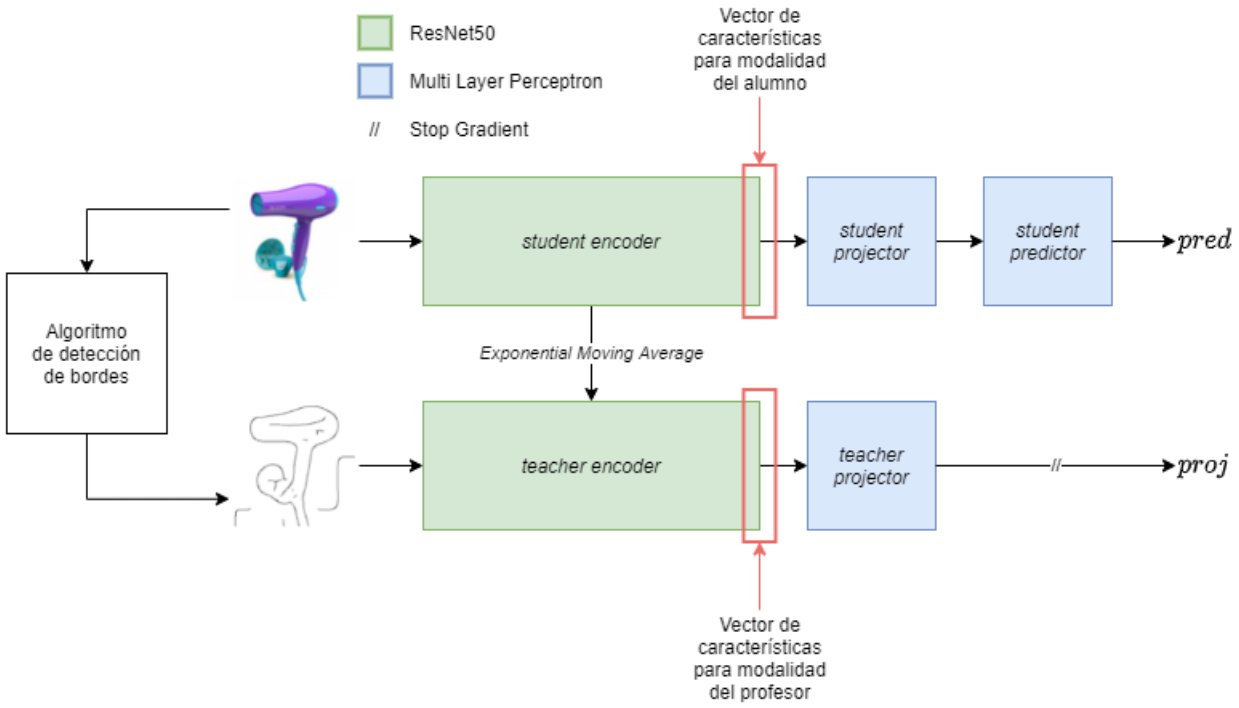


Figura 43: Idea de BYOL bimodal auto-supervisado.

Como experimento preliminar se implementa el modelo descrito y se entrena con fotos de elementos del *eCommerce* sin los dibujos respectivos, usando objetos distintos a los del conjunto de evaluación. Para generar un pseudo dibujo, se utiliza el algoritmo de Canny, un algoritmo para detección de bordes conocido y que se puede encontrar implementado en distintas librerías de procesamiento de imágenes. Para evaluar el modelo se utiliza el conjunto de evaluación de *eCommerce*, en la Tabla 19 se puede observar que los resultados son peores que los obtenidos con el BYOL bimodal entrenado con *Sketchy*, pero son muy cercanos a los obtenidos por una red siamesa entrenada con múltiples etapas también entrenado con *Sketchy*. La idea del modelo era que al poder entrenar con un conjunto de fotos más parecidas a las del conjunto de evaluación, entonces las métricas mejorarían. Esto no sucede en este caso, es

posible que las métricas no sean buenas debido a que el algoritmo de Canny [3] no siempre produce buenos bordes como se ve en la Figura 44. Para continuar el trabajo se podrían explorar formas más modernas para el problema de detección de bordes, hoy en día existen modelos de redes neuronales para este problema que muestran buenos resultados, por ejemplo **pidinet** [18] es un modelo reciente que es capaz de detectar bordes en imágenes con fondos ruidosos.

	mAP
BYOL bimodal	0,178
Red siamesa entrenada con multiples etapas	0,145
BYOL bimodal auto-supervisado	0,151

Tabla 19: BYOL bimodal auto-supervisado evaluado en el conjunto *eCommerce* vs. modelos entrenados con pares de *Sketchy*.

Figura 44: Ejemplo de aplicar algoritmo de Canny, a la izquierda se encuentra el objeto original y a la derecha las transformaciones.

7. Bibliografía

[1] Aníbal Fuentes Jara. Recuperación de imágenes basada en dibujos mediante redes convolucionales. 2020. URL http://repositorio.uchile.cl/handle/2250/175585.

[2] José M. Saavedra and Camila Álvarez. Deepsbir: Sketch based image retrieval using deep features. 2016.

[3] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[4] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese"time delay neural network. In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann, 1994. URL https://proceedings.neurips.cc/paper/1993/file/288cc0ff022877bd3df94bc9360b9c5d-Paper.pdf.

[5] Yonggang Qi, Yi-Zhe Song, Honggang Zhang, and Jun Liu. Sketch-based image retrieval via siamese convolutional neural network. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 2460–2464, 2016. doi: 10.1109/ICIP.2016.7532801.

[6] Tu Bui, Leonardo Ribeiro, Moacir Ponti, and John Collomosse. Sketching out the details: Sketch-based image retrieval using convolutional neural networks with multi-stage regression. *Computers & Graphics*, 71:77–87, 2018.

[7] Pablo Torres and Jose M. Saavedra. Compact and effective representations for sketch-based image retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 2115–2123, 6 2021.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017. ISSN 0001-0782. doi: 10.1145/3065386. URL https://doi.org/10.1145/3065386.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[10] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

[11] C. Dong, T. Xue, and C. Wang. The feature representation ability of variational auto-encoder. In *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pages 680–684, 2018.

[12] I. Higgins, L. Matthey, A. Pal, Christopher P. Burgess, Xavier Glorot, M. Botvinick, S. Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

[13] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Riche-

mond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. *CoRR*, abs/2006.07733, 2020. URL `https://arxiv.org/abs/2006.07733`.

[14] Diederik P. Kingma, Danilo Jimenez Rezende, Shakir Mohamed, and Max Welling. Semi-supervised learning with deep generative models. *CoRR*, abs/1406.5298, 2014. URL `http://arxiv.org/abs/1406.5298`.

[15] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. The sketchy database: Learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (proceedings of SIGGRAPH)*, 2016.

[16] Tu Bui, Leonardo Ribeiro, Moacir Ponti, and John Collomosse. Compact descriptors for sketch-based image retrieval using a triplet loss convolutional neural network. *Computer Vision and Image Understanding*, 164:27–37, 2017. ISSN 1077-3142. doi: http://dx.doi.org/10.1016/j.cviu.2017.06.007. URL `http://www.sciencedirect.com/science/article/pii/S1077314217301194`.

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

[18] Zhuo Su, Wenzhe Liu, Zitong Yu, Dewen Hu, Qing Liao, Qi Tian, Matti Pietikäinen, and Li Liu. Pixel difference networks for efficient edge detection. *CoRR*, abs/2108.07009, 2021. URL `https://arxiv.org/abs/2108.07009`.

8. Anexo

8.1. Cantidad de dimensiones del espacio latente de un VAE

En la Figura 45 se puede ver que la calidad de la reconstrucción mejora cuando se aumenta el número dimensiones del espacio latente. Con bajas dimensiones el modelo aún es capaz de generar una forma con algunas características del dibujo original, aunque con bastante ruido y manchas al interior. A medida que se duplica la cantidad de dimensiones se ve como comienzan a aparecer más detalles en la reconstrucción. Sin embargo, al evaluar la efectividad de los vectores del espacio latente para realizar búsquedas, como se ve en la Tabla 20, se encuentra que los vectores de baja dimensionalidad presentan mejores resultados, a pesar de lograr peores reconstrucciones. Este resultado es similar a lo que encuentra Pablo Torres y José Saavedra en su trabajo de representaciones compactas [7], en donde mejoran los resultados de búsquedas reduciendo los vectores a tamaños tan bajos como 4 y 8. Si bien un VAE no es capaz de reconstruir dibujos con vectores tan pequeños, es interesante como se repite el mismo concepto, este fenómeno puede significar que al restringir la cantidad de información extraída el modelo logra extraer características más relevantes.

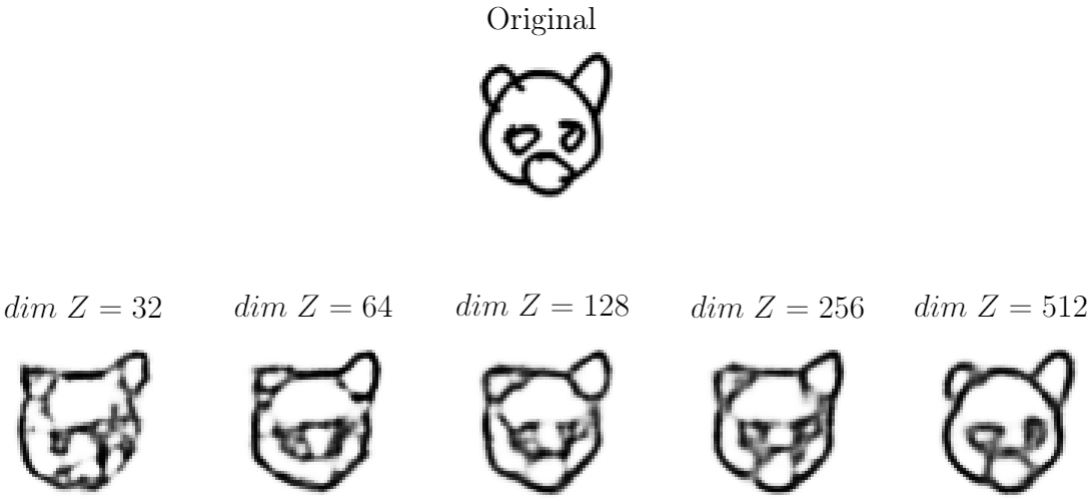


Figura 45: Efecto de cambiar el tamaño del espacio latente en la reconstrucción.

	knn accuracy	**map@5**
VAE $dim\ Z = 512$	0,147	0,136
VAE $dim\ Z = 256$	0,175	0,160
VAE $dim\ Z = 128$	0,224	0,200
VAE $dim\ Z = 64$	0,326	0,289
VAE $dim\ Z = 32$	**0,338**	**0,307**

Tabla 20: Efecto en métricas de cambiar el número de dimensiones del espacio latente.

8.2. Ajuste del parámetro β de un VAE

El efecto del peso β de la pérdida KLD indica que tanto se normaliza el espacio latente que genera el modelo VAE. En el contexto que búsquedas de dibujos, se ve que valores bajos de β mejoran la calidad de los vectores de características producidos, como se ve en la Tabla 21, en particular un valor de 0,1 parece mostrar los mejores resultados sin revertir el modelo a un Autoencoder regular, como se ve en la Figura 46 donde se generan imágenes intermedias entre dos dibujos. Esto tiene sentido ya que valores de β más altos generan un espacio con una pérdida de KLD más baja que se acerca más a una distribución normal, en donde se ocupa todo el espacio generado, pero para poder diferenciar clases unas de otras es conveniente que se formen algunos sectores más agrupados en el espacio latente.

	knn accuracy	**mAP@5**
VAE $\beta = 1$	0,313	0,281
VAE $\beta = 0{,}7$	0,318	0,287
VAE $\beta = 0{,}5$	0,319	0,287
VAE $\beta = 0{,}3$	0,326	0,297
VAE $\beta = 0{,}1$	**0,338**	**0,307**

Tabla 21: Efecto en métricas de cambiar parámetro β.

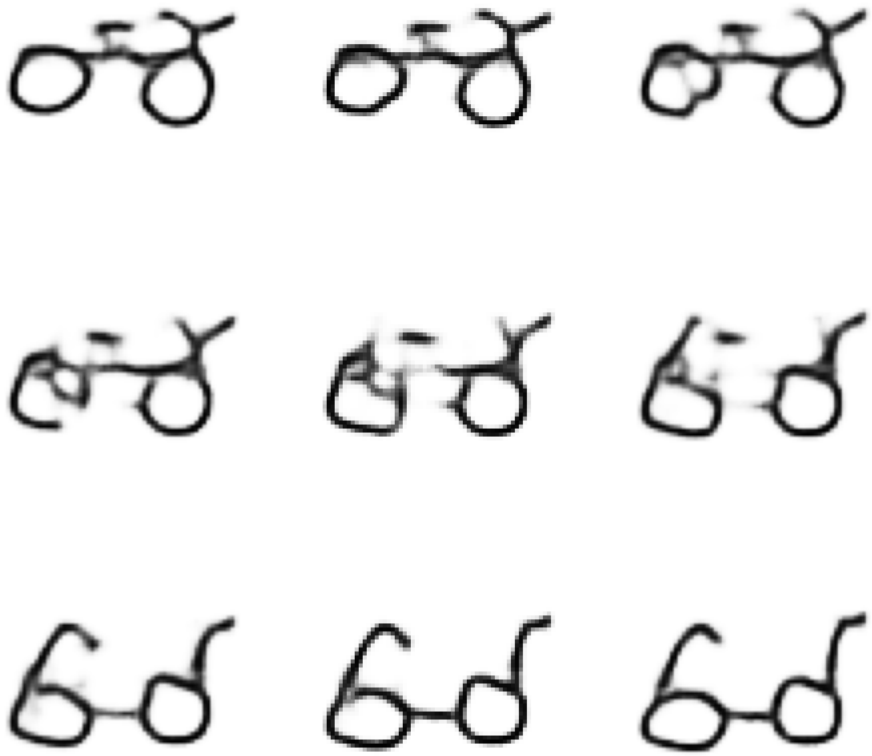


Figura 46: Efecto de recorrer el espacio latente entre dos vectores de características.

8.3. Remover lineas como transformación para dibujos en modelo BYOL

En la Tabla 22 se busca si la transformación de remover el 10 % de las líneas de un dibujo mejora los resultados obtenidos en un modelo BYOL. El modelo que no remueve líneas usa un conjunto de transformaciones básicas que incluye: rotaciones, recortes y *flips* horizontales. El modelo que si remueve líneas utiliza el mismo conjunto base de transformaciones, además de la transformación a estudiar. Claramente, el utilizar esta transformación si beneficia al modelo, mejorando los resultados las métricas en un 5 %.

	knn accuracy	**mAP@5**
BYOL sin remover líneas	0,440	0,405
BYOL con remover líneas	0,494	0,459

Tabla 22: Resultados de evaluación de modelo BYOL con y sin remover 10 % de las lineas del dibujo en el conjunto *QD eval other.*

8.4. Uso de pesos pre-entrenados en BYOL

Al introducir pesos de una ResNet 50 pre-entrenada con el conjunto ImageNet en ambas ramas de BYOL antes de entrenar, se obtiene una mejora considerable en las métricas de búsqueda de dibujos. Como se muestra en la Tabla 23, la red de ResNet 50 pre-entrenada ya tiene cierta capacidad de extracción de características de dibujos, pero al entrenar BYOL con estos pesos pre-entrenados se logra subir las métricas en alrededor de un 15 %. Esto puede estar relacionado a que BYOL busca predecir a la salida de la *target network*, que en general se inicializa de forma aleatoria, por lo que dar esta red pesos pre-entrenados puede lograr que BYOL se salte la primera etapa en la converge desde pesos aleatorios y se encamine a una mejor representación.

	knn accuracy	**mAP@5**
ResNet 50 pretrained	0,143	0,137
BYOL	0,494	0,459
BYOL + ResNet 50 pretrained	0,634	0,597

Tabla 23: Resultados de evaluación de modelo BYOL sin y con pesos pre-entrenados en el conjunto *QD eval other.*

8.5. Elección de roles en BYOL bimodal

El modelo de BYOL bimodal propuesto considera una red alumno que aprende de una red profesor, donde la segunda también aprende de la primera pero de forma mucho más lenta. Para elegir cual dominio de imágenes funciona mejor como la red profesor se entrenan dos modelos cambiando los roles. En la tabla 24 se puede ver que utilizar a la red de dibujos

como profesor logra mejores resultados, lo que tiene sentido pues las fotos contienen mucha información que no se puede extraer de un dibujo. Los resultados son distintos a los finales, pues este experimento se realiza antes de ajustar los parámetros del entrenamiento.

	knn accuracy	**mAP**
BYOL bimodal dibujos como profesor	0,441	0,187
BYOL bimodal fotos como profesor	0,158	0,080

Tabla 24: Resultados de cambiar dominio de la red profesor en BYOL bimodal.

8.6. Efecto de transformaciones en BYOL bimodal

Para evaluar que transformaciones utilizar durante el entrenamiento de BYOL bimodal se entrenan distintos modelos que aplican solo una transformación a la vez. Los modelos utilizados están descritos en la Tabla 25, a excepción de *Random Blank Patch* que se aplica solo en dibujos, las transformaciones se aplican de forma sincronizada a tanto el dibujo como la foto, es decir, ambas rotan hacia el mismo lado y los recortes se realizan en el mismo lugar. Los resultados, en la Tabla 26 muestran que aplicar cualquiera de las transformaciones propuestas los valores de las métricas empeoran, esto se puede atribuir a que como el dominio de los dibujos y el de las fotos es tan diferente, que agregar transformaciones sobre esto logra que se pierda la semántica en el par.

Transformación	**Probabilidad de aplicación**	**Descripción**
Random Blank Patch	0.5	Se sobrepone un parche circular blanco con diámetro igual a 30 % del lado de la imagen original
Random Rotation	0.5	Se rota el dibujo con un ángulo elegido entre -30° y 30°
Random Horizontal Flip	0.5	Se voltea la imagen horizontalmente
Random Sized Crop	0.5	Se hace un recorte cuadrado de la imagen en un lugar aleatorio, y con lado igual al 70 % del lado de la imagen original

Tabla 25: transformaciones utilizadas para modelo BYOL con dibujos.

	knn accuracy	**mAP**
BYOL bimodal sin transformaciones	0,532	0,311
BYOL bimodal con Random Blank Patch	0,474	0,289
BYOL bimodal con Random Rotation	0,426	0,241
BYOL bimodal con Random Crop	0,410	0,246
BYOL bimodal con Random Horizontal Flip	0,486	0,291

Tabla 26: Resultados de cambiar dominio de la red profesor en BYOL bimodal.