# Image Retrieval System based on a Binary Auto-Encoder and a Convolutional Neural Network

Andrés Ferreyra-Ramírez, Eduardo Rodríguez-Martínez, Carlos Avilés-Cruz and Fidel López-Saca

***Abstract*—The amount of image content on the Internet has increased dramatically in recent years; its precise search and retrieval is a challenge at present. The methods that have shown high efficiency are those based on convolutional neural networks (CNN) and, particularly, binary coding methods based on hashing functions. This article presents a new image retrieval scheme based on attributes from a CNN, an efficient low-dimensional binary auto-encoder, and, finally, a near-neighbor retrieval stage. The proposed methodology was tested with two image datasets CIFAR-10 and MNIST. The results are compared with existing methods in the literature.**

***Index Terms*—Binary auto-encoder, CBIR, hash, convolutional neural networks.**

## I. Introducción

EL crecimiento exponencial del uso de imágenes en la web ha generado nuevos retos, tales como su almacenamiento, su gestión y sobre todo, su recuperación [1]. La recuperación de imágenes en la web se ha realizado tradicionalmente usando meta-datos de la misma o directamente el nombre del archivo [2], [3], lo cual proporciona gran cantidad de imágenes erróneas o que no tienen nada que ver con la imagen de búsqueda. Por otro lado, existen los métodos de recuperación de imágenes por su contenido (CBIR, por sus siglas en inglés). En los sistemas CBIR se proporciona una imagen de consulta o búsqueda y el sistema regresa las imágenes más relevantes por su similitud en contenido con la de consulta [4], [5]. Existen dos clasificaciones principales de las técnicas CBIR, la basada en atributos locales y/o globales [6], [7], [8], y aquellas basadas en redes neuronales [9], [10], [11], [12]. Nuestro trabajo pertenece a la segunda clasificación.

La presente propuesta de recuperación de imágenes por su contenido, está basada en los métodos *hash*, así como en códigos binarios. Los métodos *hash* (Hash Methods) involucran técnicas de búsqueda aproximada de vecinos más cercanos. Las funciones hash permiten proyectar atributos de alta dimensión a códigos binarios de baja dimensión.

Considerando como aprenden los métodos hash, estos métodos se pueden clasificar en dos categorías, por un lado, los métodos tradicionales basados en sensibilidad local [13], [14] y aquellos basados en aprendizaje de funciones hash no lineales [15], [16]. Los primeros usan procesos de cuantificación de códigos binarios umbralizados, proyectando datos de alta dimensionalidad a espacios de menor dimensionalidad. En cuanto a los segundos, los métodos basados en aprendizaje profundo, se aprende al mismo tiempo el tipo de imagen y su función *hash*. Nuestra propuesta versa en la segunda categoría.

En este artículo, proponemos un método *hash* no supervisado para aplicaciones CBIR, que combina las ventajas de las CNN para la extracción de características y la indexación de funciones *hash* realizada por un autoencoder binario. De modo general, el sistema CBIR propuesto está constituido de dos partes principales. La primera extrae atributos de una CNN (ResNet50) entrenada con ImageNet y ajustada con las dos bases de imágenes usadas en la experimentación: CIFAR-10 y MNIST. Después se desarrolla el sistema de codificación binario, donde utiliza la indexación de funciones hash. Los experimentos realizados muestran, que el método propuesto tiene exactitud de búsqueda superior sobre métodos *hash* supervisados existentes en la literatura basados en CNN.

El resto de este documento esta organizado de la siguiente manera. El autoencoder usado es definido en la sección II. La sección III introduce la metodología propuesta para CBIR. La sección IV introduce los conjuntos de bases de imágenes utilizadas. La sección V muestra los resultados experimentales. Finalmente, la sección VI muestra las conclusiones del trabajo.

## II. Autoencoder binario propuesto

El autoencoder binario (AB) tiene como tarea el asociar atributos obtenidos de imágenes a vectores binarios por medio de funciones *hash*. Como se puede apreciar en la Fig. 1(a) el autoencoder está constituido por tres módulos principales: 1) un módulo de convolución, basado en CNN, 2) un módulo codificación binaria por medio de funciones *hash*, y por último, 3) un módulo de deconvolución, el cual hace la reconstrucción de las imágenes, basado en CNN.
El módulo 2 de los tres módulos descritos anteriormente, corresponde al módulo de codificación binaria [17]. Como se puede apreciar en la Fig. 1(b), está constituido por una red neuronal de 3 capas: una capa de entrada, una capa oculta y una capa de salida. Como es autoencoder, las capas de entrada y salida contienen el mismo número de neuronas. La capa oculta es la que realiza la codificación a $L$ bits. El autoencoder opera en modo auto-asociativo, lo que implica que los mismos atributos se presentan tanto en la entrada como en la salida de red neuronal.

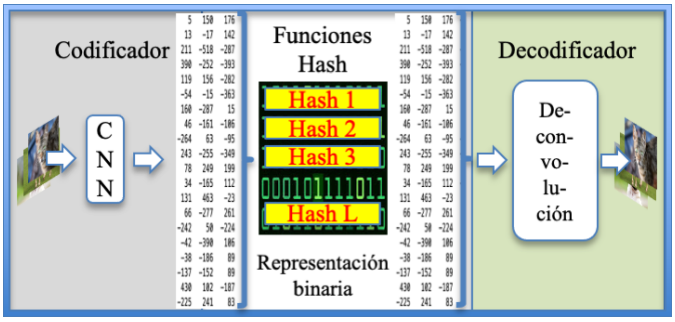
La red consta de dos partes, un codificador que genera la función $Z = h(X)$ y un decodificador que produce una

Andrés Ferreyra-Ramírez, Eduardo Rodríguez-Martínez y Carlos Avilés-Cruz trabajan en el Departamento de Electrónica de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, Av. San Pablo 180, Ciudad de México, México.{fra, erm, caviles}@azc.uam.mx.

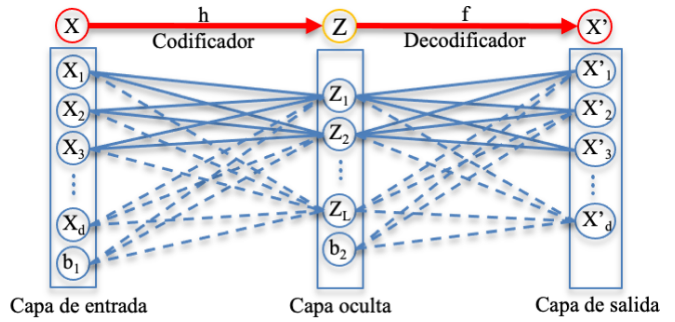
Fidel López-Saca estudió la Maestría en Ciencias de la Computación, División de Ciencias Básica e Ingeniería, Universidad Autónoma Metropolitana-Azcapotzalco, Ciudad de México, México. fidelosmcc@gmail.com.

Manuscript received March 12, 2020

reconstrucción $X' = f(Z)$.




(a)




(b)

Fig. 1: Esquema general del sistema propuesto: (a) Diagrama a bloques y (b) detalle del autoencoder binario.

Para las funciones hash, el codificador mapea las entradas continuas $X \in \mathbb{R}^d$ en vectores de código binario con $L$ bits, $Z \in \{0,1\}^L$, como sigue:

$$Z = h(X) = \phi_1(W \cdot X + b_1) \tag{1}$$

donde la función $\phi_1$ es de tipo escalón definida por la ecuación (2).

$$\phi_1(W \cdot X + b_1) = \begin{cases} 0 & \text{si } (W \cdot X + b_1) < 0 \\ 1 & \text{si } (W \cdot X + b_1) \geq 0 \end{cases} \tag{2}$$

La matriz $W$ es de pesos y es definida en el espacio $W \in \mathbb{R}^{L \times d}$, y $b_1$ es un vector de bias que es utilizado para fijar el umbral a 0 en cada bit. Después, el decodificador mapea la representación del codificador $Z$ de nuevo en un estimado del vector original, en un esfuerzo para reconstruir X desde los códigos producidos por la función hash, como sigue:

$$X' = f(Z) = \phi_2(W' \cdot Z + b_2) \tag{3}$$

donde $\phi_2$ es la función de activación del decodificador, $W' \in \mathbb{R}^{d \times L}$ es una matriz de pesos y $b_2$ es un vector de bias.

Para un conjunto de datos de patrones de alta dimensión $\chi = \{X^1, \cdots, X^N\}$, el AB es entrenado para encontrar las funciones hash $h$, minimizando la función de error (mínimos cuadrados) siguiente:

$$E_{AB}(h,f) = \sum_{n=1}^{N} ||X^n - f(h(X^n))||^2 \tag{4}$$
$$\texttt{sujeto a} \quad h(X^n) \in \{0,1\}^L$$

Para minimizar la función objetivo se utiliza un método de coordenadas auxiliares (Method of Auxiliary Coordinates MAC[1], por sus siglas en inglés) que reformula la optimización al alternar dos pasos más fáciles: uno que aprende el codificador y el decodificador por separado, y uno que optimiza el código para cada patrón de entrada. Considerando la ecuación (4) como el problema anidado, donde el modelo es $f(h)$, se introduce como coordenadas auxiliares las salidas de h; es decir, los códigos para cada uno de los $N$ patrones de entrada, obteniendo el problema con restricciones de igualdad siguiente:

$$\min_{h,f,z} \sum_{n=1}^{N} ||X^n - f(Z^n)||^2 \tag{5}$$
$$\texttt{sujeto a} \quad Z^n = h(X^n) \in \{0,1\}^L,\ n = 1, \ldots, N.$$

Aplicado el método de penalización cuadrática, en el cual los términos de penalización son los cuadrados de las violaciones de restricción, se obtiene la función objetivo mostrada en la ecuación (6); en donde $\mu$ es el parámetro de penalización. La función objetivo es minimizada a medida que $\mu$ se incrementa progresivamente, penalizando las violaciones gradualmente, de modo que las restricciones finalmente se satisfacen.

$$E_Q(h,f,Z,\mu) = \sum_{n=1}^{N} (||X^n - f(Z^n)||^2 + \mu\, ||Z^n - h(X^n)||^2)$$
$$\texttt{sujeto a} \quad Z^n \in \{0,1\}^L,\ n = 1, \ldots, N. \tag{6}$$

Para minimizar $E_Q(h,f,Z,\mu)$ se aplica optimización alterna sobre $Z$ y $(h,f)$. El algoritmo aprende las funciones $h$ (hash) y $f$ (decodificador), dados los códigos actuales, y aprende los códigos $Z$ de los patrones dadas las funciones $h$ y $f$; lo que da como resultado los siguientes dos pasos:

1) **Sobre** $(h,f)$ **para** $Z$ **fija**, se obtienen $L+1$ problemas independientes para cada una de las $L$ funciones hash de un solo bit (que intentan predecir $Z$ de manera óptima a partir de $X$), y para $f$ (que intenta reconstruir $X'$ de manera óptima desde $Z$).

2) **Sobre** $Z$ **para** $(h,f)$ **fijo**, el problema se separa para cada uno de los $N$ códigos. El vector de código óptimo para el patrón $X^n$ intenta estar cerca de la predicción $h(X^n)$ al tiempo que reconstruye $X'^n$.

[1]MAC tiene como objetivo romper las relaciones funcionales anidadas introduciendo variables como restricciones de igualdad, que son resueltas optimizando una función de penalización utilizando optimización alterna sobre los parámetros originales y las coordenadas.

Para optimizar $E_Q(h, f, Z, \mu)$ sobre $f$ para $Z$ fija, se plantea un codificador lineal de la forma $f(Z^n) = AZ^n + b$, ver ecuación (7).

La solución de la ecuación (7) se puede llevar a cabo por medio de una regresión lineal dependiente de las variables $(Z, X)$.

$$\min_f \sum_{n=1}^{N} ||X^n - f(Z^n)||^2 = \min_{A,b} \sum_{n=1}^{N} ||X^n - AZ^n - b||^2 \quad (7)$$

Para optimizar $E_Q(h, f, Z, \mu)$ sobre $h$ para $Z$ fija (sobre el codificador dados los códigos), donde $h$ es una función hash lineal de la forma $h(X) = \sigma(W \cdot X)$, $E_Q$ tiene la forma siguiente:

$$\begin{aligned}\min_h \sum_{n=1}^{N} ||Z^n - h(X^n)||^2 &= \min_W \sum_{n=1}^{N} ||Z^n - \sigma(WX^n)||^2 \\ &= \sum_{l=1}^{L} \min_{w_l} \sum_{n=1}^{N} (Z^{n,l} - \sigma(W_l^T X^n))^2\end{aligned} \quad (8)$$

Ya que la función objetivo es el número de patrones mal clasificados, se separa para cada bit $l = 1 \dots L$. Entonces, el sub problema para cada bit es un problema de clasificación binario con datos $(X, Z_l)$, usando como etiquetas las coordenadas auxiliares y el número de patrones mal clasificados como función de perdida. En esta propuesta, el sub problema de clasificación binaria, se resolvió por medio de una máquina de soporte vectorial (SVM, por sus siglas en inglés) lineal.

Para optimizar $E_Q(h, f, Z, \mu)$ con respecto a la variable $Z$ bajo la condición de las variables $h$ y $f$ fijas, conduce a una optimización binaria en el espacio $N \times L$. La optimización puede ser realizada separando en $N$ optimizaciones independientes, cada una en L variables. En consecuencia, la función a optimizar es representada en la ecuación (9), donde se puede apreciar que la minimización del error es con respecto a la variable Z.

$$\begin{aligned}&\min_Z e(Z) = ||X - f(Z)||^2 + \mu||Z - h(X)||^2 \\ &\texttt{sujeto a} \quad z_n \in \{0, 1\}^L\end{aligned} \quad (9)$$

Por lo tanto, aunque el problema sobre cada $Z^n$ es binario y $NP$ completo, se puede obtener una solución buena o incluso exacta, porque los valores prácticos de $L$ son pequeños (típicamente de 8 a 32 bits).

## III. METODOLOGÍA

La arquitectura completa del método propuesto de recuperación de imágenes por su contenido basado en descriptores provenientes de una red neuronal convolucional, así como usando codificación binaria, se muestra en la Fig. 2 y, está compuesta de dos etapas: (a) Entrenamiento y (b) Recuperación. A continuación se proporciona una descripción de las dos etapas.

### *(A) Etapa de entrenamiento*

La etapa de entrenamiento (ver parte izquierda-azul de la Fig. 2) comprende las tareas de lectura de imágenes, extracción de atributos (a través de una red neuronal convolucional) y, el aprendizaje de las funciones *hash*. Como resultado, cada imagen es representada e indexada por un vector binario, obteniendo así, una base de datos binaria indexada que contiene el código binario de cada imagen.

1) **Red neuronal convolucional**
La CNN utilizada es ResNet50 [18] tiene una profundidad de 16 bloques residuales, formados por un total de 136 capas del tipo: convolution (Conv), Batch normalization (BN) y Rectified Linear Unit (ReLU), 12 bloques con una arquitectura [Conv, BN, ReLU, Conv, BN, ReLU, Conv, BN] y los 4 bloques restantes con [Conv, BN, ReLU, Conv, BN, ReLU, Conv, BN, Conv, BN]. La capa de clasificación Softmax maneja $1,000$ clases. ResNet50 procesa imágenes de tamaño $224 \times 224 \times 3$. ResNet50 ha sido pre-entrenada con un subconjunto de datos de ImageNet [19] que contiene más de 1.2 millones de imágenes categorizadas en $1,000$ clases.
Se realizó un ajuste fino de la red para tomar en cuenta las 2 bases de imágenes utilizadas en nuestra experimentación. Para realizar el ajuste, las últimas tres capas de la red fueron reemplazadas por tres capas similares y la última capa completamente conectada nueva se adaptó al número de clases de cada uno de los conjuntos de datos utilizados. Finalmente, los atributos que serán usados por el autoencoder, provienen de la última capa de extracción de características (justo antes de la capa de clasificación) de la red ResNet50 ajustada finamente. Los $2,048$ atributos profundos son tomados.

2) **Autoencoder binario**
El autoencoder binario neuronal está constituido por tres capas (entrada-oculta-salida). Tanto la capa de entrada como la de salida están constituidas por $2,048$ neuronas. La capa oculta (capa de representación de funciones *hash*) está conformada por $L$ número de bits, siendo $L$={12, 24, 32 o 48} bits. Los códigos binarios *hash* son aprendidos mediante indexación, en donde el autoencoder binario realiza asociaciones con funciones *hash* entre los atributos de entrada y salida.

### *(B) Etapa de recuperación*

En la tarea de recuperación de imágenes (ver parte derecharoja de la Fig. 2), se realiza un proceso de búsqueda, en el cual una imagen de consulta se pasa por la CNN para extraer sus características. Una vez obtenidos los atributos, se pasan por el codificador binario y las funciones *hash*, obteniendo un código binario de indexado. El código binario es, entonces, utilizado para calcular las similitudes con los índices de las imágenes en la base de datos binaria (definida en la fase de entrenamiento). Como consecuencia, las imágenes con los puntajes más altos son devueltas. Para realizar la búsqueda más próxima se utiliza el algoritmo ANNA (Aproximate Nearest

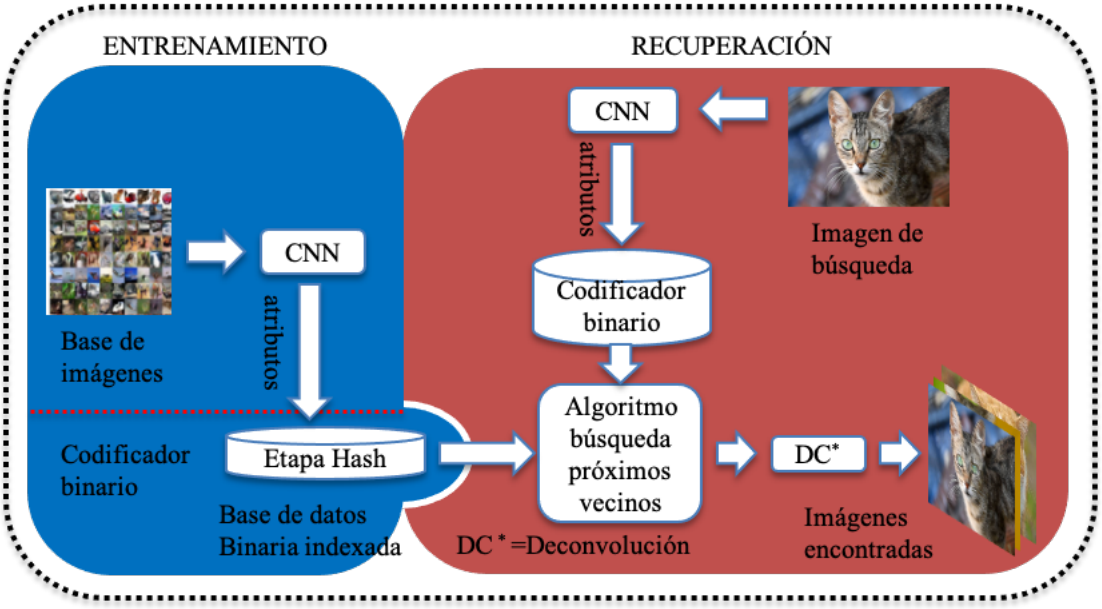



Fig. 2: Arquitectura del método CBIR propuesto.

Neighbor Algorithm) y las imágenes se recuperan mediante un proceso de indexado.

## IV. Bases de imágenes usadas

En esta sección se presentan las dos bases de imágenes usadas en la experimentación, CIFAR-10 [20] y MNIST [21]. Bases de datos ampliamente usadas como *benchmark*, por la comunidad de CBIR.

- **CIFAR-10** [20] es una base de imágenes de tamaño pequeño, son imágenes de $32 \times 32$ pixeles. Está comprendida de 10 clases y en total está constituida de $60,000$ imágenes. El total de imágenes está dividido en $50,000$ imágenes para entrenamiento ($5,000$ por clase) y $10,000$ para prueba (mil por clase). Para nuestra experimentación, seleccionamos aleatoriamente $1,000$ imágenes por clase para el entrenamiento, en total $10,000$. Para la prueba o consulta se tomaron 100 imágenes por clase aleatoriamente, en total $1,000$. A título de ejemplo, la Fig. 3(a) muestra algunas imágenes tomadas aleatoriamente.

- **MNIST** [21] está compuesta de 10 dígitos (10 clases) escritos a mano [0, 1, 2, 3, 4, 5, 6, 7, 8 y 9]. La Fig. 3(b) muestra ejemplo de las imágenes tomadas aleatoriamente. La base de imágenes está conformada de $60,000$ y $10,000$ para entrenamiento y prueba, respectivamente. Los dígitos fueron digitalizados en blanco y negro a un tamaño de $28 \times 28$ pixeles. Al igual que CIFAR-10, para la comparación con otros trabajos, se utilizan $1,000$ imágenes muestreadas aleatoriamente del conjunto de prueba, 100 por clase, para la evaluación del rendimiento.

(a) (b)

Fig. 3: Ejemplo de algunas imágenes de las dos bases usadas: (a) CIFAR-10 y (b) MNIST .

## V. Resultados experimentales

Una vez definida la metodología y las bases de imágenes a usar, se procedió a evaluar la metodología propuesta. La implementación se hizo en una estación de trabajo con 64 GB de memoria con sistema operativo Windows. Se hizo usó de una unidad de procesamiento gráfico GPU (por sus siglas en inglés) NVIDIA GeForce GTX 1080 con $2,560$ núcleos y 8 GB de memoria. Con respecto al software usado, se utilizó el Toolbox de Deep Learning de MatLab 2018b.

Para medir la eficiencia de sistema propuesto, se emplearon las métricas de *Precisión* ("Precision", ver ecuación (10)) y *Recuperación* ("Recall", ver ecuación (11)). Por otro lado, también se usó la métrica de *Precisión media promedio* (MAP, por sus siglas en inglés, ver ecuación (12)).

$$P = \frac{\text{Número de imágenes relevantes recuperadas}}{\text{Número total imágenes recuperadas}} \quad (10)$$

$$R = \frac{\text{Número de imágenes relevantes recuperadas}}{\text{Número total imágenes de la base de imágenes}} \quad (11)$$

$$MAP = \frac{\sum_{i=1}^{Q} \bar{P}_i}{\text{Q}} \quad (12)$$

donde $Q$ es el número de imágenes buscadas y $\bar{P}_i$ el valor medio de la Precisión para la evaluación $i$.

### A. Evaluación en entrenamiento

La propuesta está comprendida de una fase de entrenamiento y otra de prueba. Inicialmente, se hace la evaluación de toda la fase de entrenamiento, la cual comprende el entrenamiento fino de ResNet50 con las dos bases de imágenes y el entrenamiento del codificador binario por separado. El objetivo que persigue el entrenamiento del codificador binario es la obtención de las funciones *hash* y la evaluación de la longitud de los códigos binarios (número de bits). La parte de prueba se lleva a cabo con imágenes que no fueron utilizadas en el entrenamiento. A continuación se describe cada una de las fases.

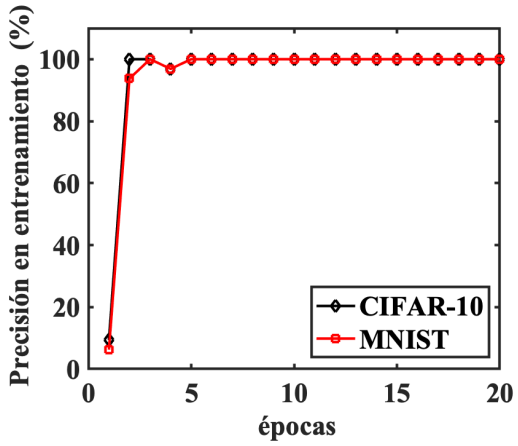
- **Entrenamiento de la red ResNet50**

Tomando la red ResNet50 pre-entrenada con ImageNet, se hizo un entrenamiento fino con las 2 bases de imágenes propuestas. La CNN fue ajustada utilizando el método del gradiente descendente estocástico. El tamaño de lote usado fue de 32 imágenes, con momento de 0.9 y factor de regularización (decaimiento de pesos) de 0.0001. La tasa de aprendizaje inicial fue de 0.001 y se disminuyó en un factor de 10 después de cada 5 épocas. Finalmente, se utilizó aumento de datos en la fase de aprendizaje.

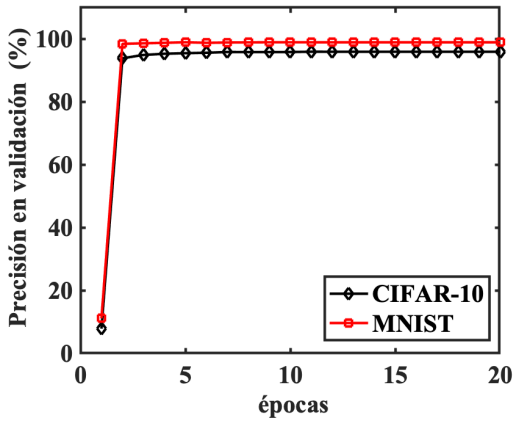
Una vez definidos los parámetros, se procedió a analizar la precisión y el error tanto en entrenamiento de la red, como en validación. La Fig. 4 muestra los resultados de la precisión para las dos bases de imágenes, se observa que a partir de 10 épocas, se llega a la zona estable y, por ende asumimos que la red ha aprendido adecuadamente.

- **Codificador Binario**

Los atributos que se extraen de la CNN ResNet50, características profundas de $2,048$ dimensiones, son utilizados para entrenar al autoencoder binario en una modalidad de aprendizaje no-supervisado. En la fase de entrenamiento del codificador binario, los atributos son utilizados como las entradas y las salidas del autoencoder, con el fin de reconstruir los atributos de entrada y con esto conservar al máximo la similitud de las imágenes que representan. Para realizar la codificación binaria, el número de neuronas de la capa oculta se define fijando el número de bits (L={12, 24, 32 o 48}) a generar. Los códigos binarios son aprendidos mediante indexación, en donde el autoencoder binario realiza asociaciones con funciones hash entre los atributos de entrada y salida. En la Fig. 1(b) se puede apreciar esquemáticamente el funcionamiento del autoencoder, en esta Fig. 1(b) se utiliza la variable X para representar los atributos de las dos bases de imágenes utilizadas.




(a)




(b)

Fig. 4: Precisión del entrenamiento de la red ResNet50 para las dos bases de imágenes usadas: (a) Precisión en entrenamiento, (b) Precisión en validación.

Para evaluar el entrenamiento del autoencoder binario, se ejecutaron 60 iteraciones del algoritmo de optimización alterna para minimizar la ecuación (6). Cada iteración consta de los dos pasos descritos en la Sección II, optimización de (h,f) para Z fija, y optimización de Z para (h,f) fijos. La Fig. 5 muestra el valor de la función objetivo después de cada iteración. Como se puede apreciar en la Fig. 5, el error mínimo se encuentra después de 20 iteraciones, es decir, el error no cambia y se mantiene constante para los 4 codificadores a 12, 24, 32 y 48 bits. Lo que indica que el proceso de codificación-decodificación ha llegado a una zona de estabilidad y podrá ser usado. Cabe remarcar que el error mínimo se obtuvo con una codificación de 12 bits (trazo rosa-circular), el error máximo se encuentra usando códigos de longitud de 48 bits.

### B. Resultados CBIR en función del número de bits del codificador

En cuanto a la evaluación del sistema propuesto, con respecto al número de bits usados en el codificador binario. La Tabla I muestra el porcentaje de éxito sobre las primeras $1,000$ imágenes devueltas por el sistema propuesto, usando

TABLA I: Resultado de la precisión media promedio (MAP, por sus siglas en inglés) de la recuperación de $1,000$ imágenes para las bases CIFAR-10 y MNIST; variando la longitud del código binario.

| Base de imágenes | **12** | **24** | **32** | **48** |
|---|---|---|---|---|
| *MNIST* | **98.16** | 97.66 | 97.51 | 97.19 |
| *CIFAR-10* | **93.84** | 92.55 | 91.99 | 92.98 |

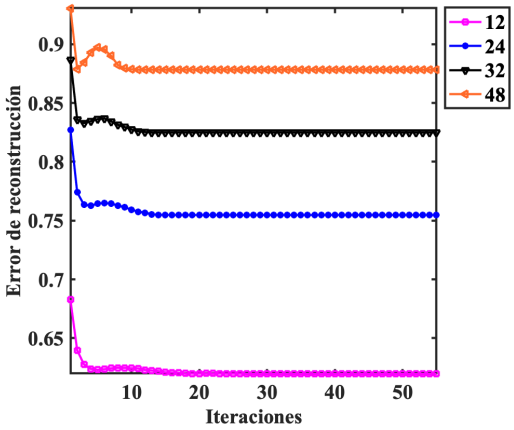



Fig. 5: Error de recuperación en el codificador binario de acuerdo a la ecuación (6).

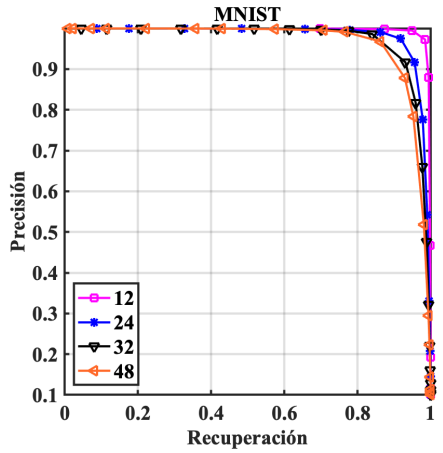
como métrica la *precisión media promedio* (MAP, por sus siglas en inglés) sobre ambas bases de imágenes, CIFAR-10 y MNIST. Como se puede apreciar en la Tabla I, el mejor MAP se encuentra a 12 bits para MNIST y CIFAR-10 a 98.16% y 93.84%, respectivamente.

La Fig. 6 muestra el resultado Precisión-Recuperación para las dos bases de imágenes usadas. Se tomaron como punto de análisis el número de bits usados en el codificador binario. Como se puede apreciar en la Fig. 6(a) para la base de imágenes MNIST, la recuperación de imágenes es excelente a 12 bits. En cuanto a los resultados obtenidos para la base de imágenes CIFAR-10 (ver Fig. 6(b)), la recuperación también es excelente, aunque ligeramente menor que para la base MNIST.

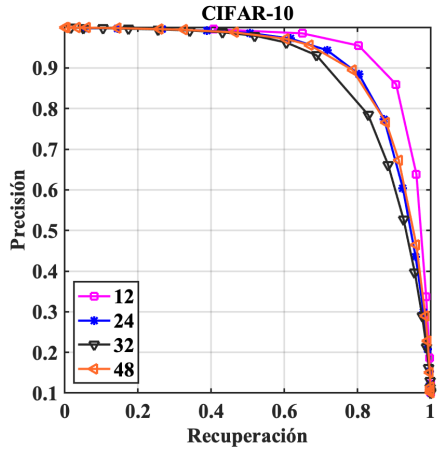
Como ejemplo de la recuperación de imágenes por su contenido, la Fig. 7 muestra la búsqueda y recuperación con la base de imágenes CIFAR-10. La imagen de búsqueda es la primera columna del lado izquierdo (Q); de la segunda a la onceava columna se presentan las imágenes encontradas (*R1, R2, R3, R4, R5, R6, R7, R8, R9, y R10*) más parecidas por el algoritmo propuesto. Como se puede apreciar en la Fig. 7, todas la imágenes recuperadas pertenecen a la misma clase de la buscada. Para las 10 clases comprendidas en la base CIFAR-10, prácticamente se recuperan exitosamente todas.

### *C. Comparación contra el estado del arte*

Comparamos el método propuesto con varios métodos hash supervisados [22], [23], [24], [25] basados en el aprendizaje profundo de última generación en los conjuntos de datos CIFAR-10 y MNIST. La Tabla II muestran los resultados de recuperación en términos de la precisión media promedio (MAP) de las primeras $1,000$ imágenes devueltas por los diferentes métodos. Los resultados en la sección superior




(a) Base de imágenes MNIST.




(b) Base de imágenes CIFAR-10.

Fig. 6: Resultado de Precisión-Recuperación para diferente número de bits usados en el codificador binario.

de la Tabla provienen de la base de imágenes CIFAR-10, mientras que los de la parte inferior corresponde a la base de imágenes MNIST.

Para la base de imágenes CIFAR-10 (ver Tabla II), el método propuesto logra una precisión de búsqueda ligeramente mayor a los métodos supervisados. Por ejemplo, en comparación con el mejor método supervisado (SSDH), los resultados del MAP del método propuesto indican un aumento relativo del 1.36% al 8.08%. Con respecto a la base de imágenes MNIST (ver Tabla II), el método propuesto logra una precisión de búsqueda muy competitiva con los métodos supervisados. Por ejemplo, en comparación con el mejor método supervisado (SSDH), los resultados del MAP del método propuesto indican una disminución relativa del

TABLA II: Comparación contra otros métodos *hash* de aprendizaje profundo reportados en la literatura. La medida usada es la precisión media promedio (MAP, por sus siglas en inglés) de la recuperación sobre $1,000$ imágenes para las bases CIFAR-10 y MNIST; variando la longitud del código binario.

| | **Método** | **12** | **24** | **32** | **48** |
|---|---|---|---|---|---|
| CIFAR-10 | CNNH [22] | 43.9 | 51.1 | 50.9 | 52.2 |
| | CNNH+ [23] | 55.2 | 56.6 | 55.8 | 58.1 |
| | Peng et. al [24] | 83.85 | 84.50 | 85.07 | 85.56 |
| | SSDH [25] | 90.59 | – | 90.63 | 91.45 |
| | **Propuesto** | **93.84** | **92.55** | **91.99** | **92.98** |
| MNIST | CNNH [22] | 95.7 | 96.3 | 95.6 | 96.0 |
| | CNNH+ [23] | 96.9 | 97.5 | 97.1 | 97.5 |
| | SSDH [25] | 99.31 | – | 99.37 | 99.39 |
| | **Propuesto** | **98.16** | **97.66** | **97.51** | **97.19** |

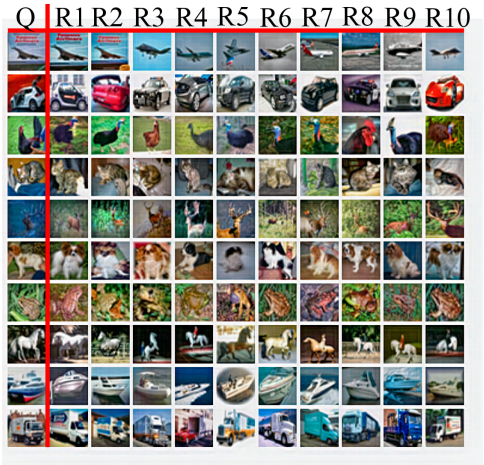



Fig. 7: Ejemplo de recuperación *top-10* de imágenes de la base CIFAR-10 a 48 bits.

1.15% al 2.21%.

## VI. Conclusiones

En este artículo, se propuso un método hash no supervisado para aplicaciones de CBIR, que combina las ventajas de las CNN para la extracción de características y la indexación de funciones hash realizada por un autoencoder binario. Los experimentos realizados muestran, que el método propuesto tiene exactitud de búsqueda superior sobre métodos hash supervisados basados en CNN reportados en la literatura.

El sistema propuesto está constituido de dos parte principales: 1) una etapa de extracción de atributos basado en redes convolucionales, y 2) un sistema de codificación binario, basado en funciones *hash*.

El sistema CBIR propuesto funcionó exitosamente en las bases de datos: CIFAR-10 y MNIST.

El algoritmo propuesto toma eficientemente atributos de la última capa de la CNN ResNet50 pre-entrenada con base de imágenes ImageNet, posteriormente se hizo un entrenamiento fino con las dos bases de imágenes experimentación.

Con respecto al algoritmo de codificación binario, éste se implementó exitosamente en modo no-supervisado. Se evaluó su desempeño en función del número de bits usados, obteniendo los mejores resultados a 12 bits (baja dimensionalidad).

La propuesta es superior en $8.08\%$ a 12 bits de codificación con respecto a la metodología más competitiva de la literatura, usando la base de imágenes CIFAR-10. Con respecto a la Base de imágenes MNIST, el método propuesto es igual de competitivo que los existente en el estado del arte.

Como trabajo futuro se analizarán más bases de imágenes. Por otro lado, también se trabajará con funciones *hash* no autoasociativas.

## References


[1] C. Yan, Y. Tu, X. Wang, Y. Zhang, X. Hao, Y. Zhang, and Q. Dai, "Stat: Spatial-temporal attention mechanism for video captioning," *IEEE Transactions on Multimedia*, vol. 22, no. 1, pp. 229–241, Jan 2020.

[2] M. Marinov, I. Valova, and Y. Kalmukov, "Comparative analysis of content-based image retrieval systems," 2019.

[3] S. Laborie, A. Manzat, and F. Sedes, "Managing and querying efficiently distributed semantic multimedia metadata collections," *IEEE Multimedia*, 2019.

[4] C. Benavides, J. Villegas, G. Roman, and C. Aviles, "Face classification by local texture analisys through cbir and surf points," *IEEE Latin America Transactions*, vol. 14, no. 5, pp. 2418–2424, 2016.

[5] S. Cervantes, A. Mexicano, J. Cervantes, R. Rodríguez, and J. Fuentes-Pacheco, "Binary pattern descriptors for scene classification," *IEEE Latin America Transactions*, vol. 18, no. 01, pp. 83–91, 2020.

[6] S. Wang, K. Han, and J. Jin, "Review of image low-level feature extraction methods for content-based image retrieval," *Sensor Review*, vol. 39, no. 6, pp. 783–809, 2019.

[7] N. Ghosh, S. Agrawal, and M. Motwani, "A survey of feature extraction for content-based image retrieval system," *Lecture Notes in Networks and Systems*, vol. 34, pp. 305–313, 2018.

[8] R. Asery, P. Marwaha, R. Sunkaria, and L. Sharma, *Image retrieval techniques using content-based local binary descriptors: A survey*, 2017.

[9] M. Garg, M. Malhotra, and H. Singh, "Comparison of deep learning techniques on content based image retrieval," *Modern Physics Letters A*, 2019.

[10] F. Mustafic, I. Prazina, and V. Ljubovic, "A new method for improving content-based image retrieval using deep learning," 2019.

[11] N. Passalis, A. Iosifidis, M. Gabbouj, and A. Tefas, "Variance-preserving deep metric learning for content-based image retrieval," *Pattern Recognition Letters*, vol. 131, pp. 8–14, 2020.

[12] J. Salas, F. de Barros Vidal, and F. Martinez-Trinidad, "Deep learning: Current state," *IEEE Latin America Transactions*, vol. 17, no. 12, pp. 1925–1945, 2019.

[13] D. Li, W. Zhang, S. Shen, and Y. Zhang, "Ses-lsh: Shuffle-efficient locality sensitive hashing for distributed similarity search," 2017, pp. 822–827.

[14] W. Hu, Y. Fan, J. Xing, L. Sun, Z. Cai, and S. Maybank, "Deep constrained siamese hash coding network and load-balanced locality-sensitive hashing for near duplicate image detection," *IEEE Transactions on Image Processing*, vol. 27, no. 9, pp. 4452–4464, 2018.

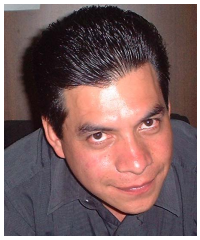
[15] J. Wang, T. Zhang, j. song, N. Sebe, and H. T. Shen, "A survey on learning to hash," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 769–790, April 2018.

[16] X. Tian, X. Zhou, W. Ng, J. Li, and H. Wang, "Bootstrap dual complementary hashing with semi-supervised re-ranking for image retrieval," *Neurocomputing*, vol. 379, pp. 103–116, 2020.

[17] M. A. Carreira-Perpinán and R. Raziperchikolaei, "Hashing with binary autoencoders," in *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, 2015, pp. 557–566.
[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.
[20] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.
[21] Y. LeCun, "The mnist database of handwritten digits," *http://yann.lecun.com/exdb/mnist/*, 1998.
[22] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in *Proceedings of the Twenty-eighth AAAI conference on artificial intelligence*, pp. 2156–2162.
[23] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3270–3278.
[24] T.-q. Peng and F. Li, "Image retrieval based on deep convolutional neural networks and binary hashing learning," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1742–1746.
[25] H.-F. Yang, K. Lin, and C.-S. Chen, "Supervised learning of semantics-preserving hash via deep convolutional neural networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 2, pp. 437–451, 2018.

**Andrés Ferreyra-Ramírez** Andrés Ferreyra Ramírez, es Ingeniero Mecánico Electricista con Especialidad en Electrónica, por la Universidad Nacional Autónoma de México. Maestro en Ingeniería Biomédica, por la Universidad Autónoma Metropolitana, unidad Iztapalapa. Doctorado en Ciencias en Control Automático, por el Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional CINVESTAV-IPN. Desde 1996 se desempeña como Profesor Investigador Titular "C" de tiempo completo en el Departamento de Electrónica de la UAM-Azcapotzalco, sus áreas de investigación son: cómputo suave, aprendizaje de máquina, sistemas de transporte inteligentes y aprendizaje profundo.

**Eduardo Rodríguez-Martínez** Eduardo Rodríguez Martínez, obtuvo el grado de Maestro en Ciencias de la Información e Inteligencia, y el de Doctor en Ciencias con especialidad en Extracción de Características, en 2008 y 2012, respectivamente, por parte de la Universidad de Liverpool, Reino Unido. Actualmente es profesor asociado en el Departamento de Electrónica de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco. Sus principales líneas de investigación son la computación evolutiva, la extracción de características, y el reconocimiento de patrones, así como aplicaciones en el campo de la robótica, y en el diseño de interfaces cerebro-computadora.



**Carlos Avilés-Cruz** Carlos Avilés Cruz, obtuvo el título de Ingeniero en Electrónica especialidad en Sistemas Digitales en la Universidad Autónoma Metropolitana-México en 1991, el grado Maestro con especialidad en procesamiento digital de señales, imágenes y voz en el Instituto Politécnico Nacional de Grenoble-Francia en 1993, y el grado de Doctor del Instituto Politécnico Nacional de Grenoble-Francia en 1997. Es autor y coautor de más de 60 artículos en revistas más congresos nacionales e internacionales, además de coautor de dos libros. Sus intereses son la visión por computadora, el procesamiento digital de imágenes, el procesamiento digital de señales, el reconocimiento de patrones y la estadística de orden superior. Es miembro del Sistema Nacional de Investigadores en México.

**Fidel López-Saca** Fidel López Saca, recibió su título de Licenciado en Matemáticas Aplicadas y Computación en 2016 en la Universidad Nacional Autónoma de México (UNAM), y el grado de Maestro en Ciencias de la Computación en 2019 por la Universidad Autónoma Metropolitana-Azcapotzalco, con la especialidad en reconocimiento de patrones y procesamiento digital de imágenes. Sus intereses de investigación son: redes convolucionales, aprendizaje profundo, visión por computadora y reconocimiento de patrones.