# Investigación
# *Autoencoders*

1st Camacho Hernández José Julián
*Instituto Tecnológico de Costa Rica*
*Reconocimiento de Patrones*
Cartago, Costa Rica
jcamacho341@estudiantec.cr

2nd Guillén Fernández José Leonardo
*Instituto Tecnológico de Costa Rica*
*Reconocimiento de Patrones*
Cartago, Costa Rica
leoguillen@estudiantec.cr

3nd Abarca Abigail
*Instituto Tecnológico de Costa Rica*
*Reconocimiento de Patrones*
Cartago, Costa Rica
sabigailab@estudiantec.cr

4nd Guzmán Quesada Joshua
*Instituto Tecnológico de Costa Rica*
*Reconocimiento de Patrones*
Cartago, Costa Rica
joshagq@estudiantec.cr

***Resumen*—Autoencoders have emerged as powerful tools in the field of unsupervised learning, offering significant potential in various domains, such as image and text data analysis. This paper presents a comprehensive exploration of autoencoders, aiming to elucidate their functionality and components through a practical example implemented in a programming environment.**

***Index Terms*—Machine Learning, Autoencoders, latent space, self-supervised**

## I. Introducción

En la era de la Inteligencia Artificial (IA), se ha observado una minimización de los esfuerzos humanos gracias a la aplicación de la automatización en diversos procesos. A medida que la tecnología avanza, también aumentan las capacidades de almacenamiento y computación. Los conceptos de Aprendizaje Profundo (Deep Learning) y Aprendizaje Automático (Machine Learning) han existido desde hace aproximadamente 60 años, pero debido a la falta de recursos adecuados, su exploración y desarrollo han sido limitados.

En los últimos años, se ha experimentado un crecimiento significativo en el campo de la IA, y esta tecnología se ha aplicado en diversos sectores como la salud, los seguros, la banca, entre otros. Este avance ha sido posible gracias a la disponibilidad de potentes Unidades de Procesamiento Gráfico (GPU) y mecanismos de almacenamiento más rápidos.

Sin embargo, es importante destacar que el almacenamiento y procesamiento eficiente de datos juegan un papel crucial en el desarrollo de la IA. Específicamente, los *autoencoders* se han vuelto prominentes en el campo del Aprendizaje Automático debido a su capacidad de compresión de datos. Los *autoencoders* son modelos auto-supervisados que pueden aprender a comprimir de manera eficiente los datos de entrada. Aunque la compresión de datos es solo uno de los casos de uso de los *autoencoders*, existen numerosas aplicaciones adicionales de esta técnica.

En este contexto, el presente estudio tiene como objetivo profundizar en el concepto de *autoencoder* y su relevancia en el campo del Aprendizaje Automático. A través de una revisión exhaustiva de la literatura y el análisis de un caso de uso, se explorarán las características, ventajas y limitaciones de los *autoencoders*. Además, se investigará cómo los mismos pueden contribuir al avance de la IA en diferentes industrias, mencionando su aplicabilidad en áreas como la salud, los seguros y la banca.

## II. Estado del Arte

***Autoencoders* básicos:** Los *autoencoders* son una clase popular de redes neuronales utilizadas para aprender representaciones eficientes de los datos de entrada. Un *autoencoder* básico consta de una capa de codificación que reduce la dimensión de los datos y una capa de decodificación que reconstruye los datos originales. La información se comprime y luego se descomprime, lo que permite capturar características esenciales de los datos. [1]

**Variantes de *autoencoders*:** Se han propuesto diversas variantes de *autoencoders* para abordar diferentes desafíos en el aprendizaje de representaciones. Algunos ejemplos incluyen los *autoencoders* Variacionales (VAE) que incorporan la teoría de inferencia variacional para generar muestras de datos latentes de manera generativa, y los Denoising *autoencoders* (DAE) que se entrenan para reconstruir datos corruptos, lo que ayuda a aprender representaciones más robustas. [2, 3]

**Aplicaciones en reconstrucción de imágenes:** Los *autoencoders* se han utilizado ampliamente en tareas de reconstrucción y generación de imágenes. El uso de *autoencoders* convolucionales ha demostrado resultados prometedores en la reconstrucción y generación de imágenes realistas. Algunos enfoques combinan *autoencoders* con técnicas de generación adversarial (GAN) para mejorar aún más la calidad de las imágenes generadas. [4]

**Reducción de dimensionalidad:** Los *autoencoders* también se han utilizado para la reducción de dimensionalidad, donde ayudan a capturar las características más importantes de los datos mientras eliminan el ruido y la redundancia. Al comprimir la información en un espacio de menor dimensión, los

*autoencoders* pueden ayudar a visualizar y comprender mejor los datos en conjuntos de alta dimensionalidad. [5]

**Detección de anomalías:** Los *autoencoders* también han demostrado ser efectivos en la detección de anomalías o anomalías en los datos. Al entrenar un *autoencoder* en datos normales, puede identificar patrones y características comunes. Luego, cuando se le presenta una instancia anómala, el *autoencoder* tiene dificultades para reconstruirla con precisión, lo que indica una anomalía. Esto se ha aplicado en campos como la detección de fraudes y la detección de intrusos. [6]

## III. Desarrollo

Un *autoencoder* es un tipo de red neuronal que se utiliza en el aprendizaje automático para la compresión y reconstrucción de datos. Se considera un modelo auto-supervisado, lo que significa que no requiere etiquetas externas para el entrenamiento, sino que utiliza los datos de entrada como su propia ”supervisión”.

La estructura básica de un *autoencoder* consta de dos partes principales: la capa de codificación (encoder) y la capa de decodificación (decoder). El *encoder* transforma los datos de entrada en una representación de menor dimensionalidad, llamada *espacio latente* o *código*. Esta representación captura las características esenciales de los datos y contiene una versión comprimida de la información original. Luego, el *decoder* toma esta representación latente y la reconstruye para producir una versión aproximada de los datos de entrada originales.

El objetivo principal de un *autoencoder* es minimizar el error de reconstrucción entre los datos de entrada y la versión reconstruida. Durante el entrenamiento, se ajustan los pesos y los sesgos de las capas del *encoder* y del *decoder* para mejorar la precisión de la reconstrucción. Al aprender a comprimir y reconstruir los datos, el *autoencoder* puede capturar patrones, reducir el ruido y eliminar la información redundante en los datos de entrada.

Además de su capacidad para comprimir y reconstruir datos, los *autoencoders* también se utilizan para otras tareas, como la reducción de dimensionalidad y la generación de datos nuevos. Las variantes de *autoencoders*, como los *autoencoders* Variacionales (VAE) y los *Denoising autoencoders* (DAE), han sido desarrolladas para abordar desafíos específicos y mejorar el rendimiento en diferentes escenarios.

En la figura 1, se presenta un diagrama que ejemplifica la arquitectura de un *autoencoder* compuesto por redes convolucionales, utilzado para el procesamiento de imágenes.

## IV. Implementación

Una implementación de *autoencoders* en la aplicación de compresión de archivos de imagen implica utilizar esta técnica de aprendizaje automático para comprimir imágenes de manera eficiente. En lugar de utilizar métodos tradicionales de compresión como JPEG o PNG, que pueden llevar a pérdida de calidad, los *autoencoders* pueden capturar y representar las características esenciales de una imagen de manera más efectiva.

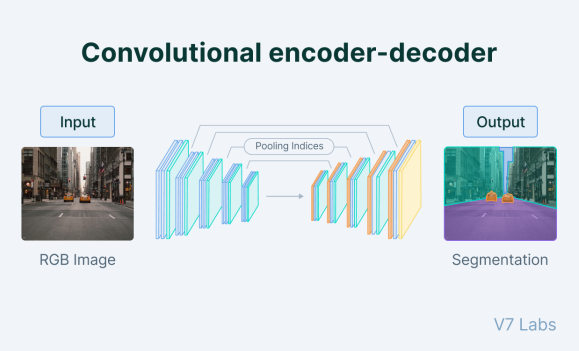


Figura 1. Diagrama de un *autoencoder*

En este enfoque, se entrena un *autoencoder* utilizando un conjunto de imágenes como datos de entrada. El *autoencoder* aprende a codificar y decodificar estas imágenes, creando una representación latente (o código) que captura las características principales de las imágenes. Luego, este código se utiliza para reconstruir una versión aproximada de la imagen original.

La clave de la compresión de imágenes con *autoencoders* radica en la capacidad del modelo para aprender una representación latente más compacta que la imagen original. Al capturar solo las características más importantes, se puede lograr una mayor compresión sin perder demasiada calidad visual en la reconstrucción. Además, los *autoencoders* pueden aplicar técnicas como la reducción de ruido y la eliminación de información redundante durante el proceso de compresión, lo que puede mejorar aún más la calidad de la imagen reconstruida.

Una vez que el *autoencoder* ha sido entrenado, se puede utilizar para comprimir imágenes nuevas. La imagen de entrada se codifica utilizando el *encoder* y se guarda la representación latente resultante. Luego, se puede almacenar esta representación de manera más eficiente en comparación con la imagen original. Para descomprimir la imagen, se utiliza el *decoder* para reconstruir la imagen a partir de la representación latente.

Para este problema relacionado con la compresión de imágenes, se plantea el caso aplicado del *set* de datos Fashion-MNIST. Este es un conjunto de datos de imágenes que consta de un conjunto de entrenamiento de 60 000 muestras y un conjunto de prueba de 10 000. Cada una de ellas se encuentra en escala de grises de $28 \times 28$, y está asociada con una etiqueta de las 10 clases, que son: [7]

1. Camiseta/top
2. Pantalón
3. Jersey
4. Vestido
5. Abrigo
6. Sandalia
7. Camisa
8. Zapatillas
9. Bolsa
10. Botín

Para implementar el caso específico, se utilizará el diagrama de la figura 2.

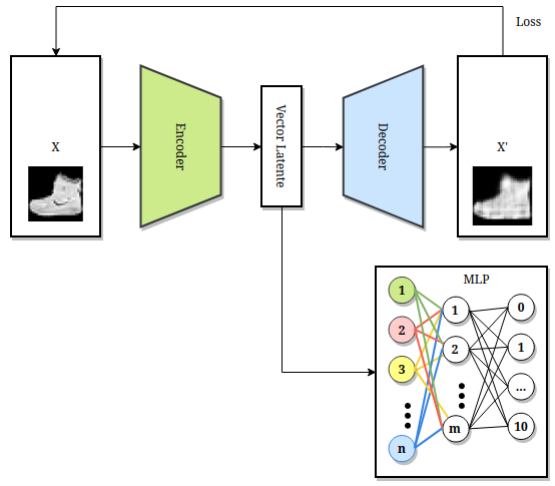


Figura 2. Diagrama de la implementación del *autoencoder*

Para implementarlo, se utiliza la biblioteca *Pytorch*. Con esta se define el modelo del *autoencoder* de la siguiente manera:

```
1  class Autoencoder(nn.Module):
2      def __init__(self, encoding_dim):
3          super(Autoencoder, self).__init__()
4
5          self.encoder = nn.Sequential(
6              nn.Linear(28*28, 128),
7              nn.ReLU(),
8              nn.Linear(128, encoding_dim)
9          )
10
11         self.decoder = nn.Sequential(
12             nn.Linear(encoding_dim, 128),
13             nn.ReLU(),
14             nn.Linear(128, 28*28),
15             nn.Sigmoid()
16         )
17
18     def forward(self, x):
19         encoded = self.encoder(x)
20         decoded = self.decoder(encoded)
21         return decoded
```

Listing 1. Implementación del autoencoder.

Del *listing* 1, se aprecian aspectos importantes de un *autoencoder*. En primer lugar, en la línea 19 se visualiza la aplicación del *encoder*. En este punto se llama al bloque de que inicia en la línea 5, donde se observa que en efecto el *encoder* es una red neuronal que recibe las imágenes de entrada.

De igual manera, se observa en la línea 11 que el *decoder* es otra red neuronal, que tiene como insumo el resultado del *encoder*. Dicho resultado es el denominado **vector latente**.

En la presente implementación, se entrenará el *autoencoder* y se utilizará para realizar predicciones y así visualizar las imágenes descomprimidas. Seguidamente, se demostrará la utilidad del vector latente, que conforma las imágenes comprimidas, en el *transfer learning*. De esa manera, se realizarán predicciones en otro modelo completamente diferente como lo es un perceptrón multicapa (MLP), usando como insumo dicho vector, tal como se visualiza en la estructura que se presentó en la figura 2.

## V. Resultados

En la presente sección se presentan los resultados esperados y obtenidos relacionados con el caso de aplicación.

### V-A. *Resultados esperados*

Se espera que el *autoencoder* comprima eficientemente las imágenes en escala de grises, preservando la información esencial. El *autoencoder* debería ser capaz de reconstruir con precisión las imágenes originales a partir de las representaciones comprimidas.

Utilizando el *autoencoder* entrenado, se pueden generar las imágenes descomprimidas a partir de las representaciones comprimidas. Estas visualizaciones pueden ayudar a evaluar la fidelidad del proceso de reconstrucción y proporcionar información sobre la información retenida en la representación comprimida.

Se puede evaluar el rendimiento del *autoencoder* en la predicción de las etiquetas asociadas (clases) de las imágenes utilizando métricas de clasificación como precisión, *recall* y *F1-score*. Se espera que el *autoencoder* pueda reconstruir con precisión y predecir las etiquetas de estas imágenes no vistas.

Asimismo, respecto al vector latente se espera que, al utilizarlo como entrada en otro modelo, sea posible entrenarlo y realizar predicciones, ya que este vector corresponde a una versión comprimida o simplificada del *set* de datos original.

### V-B. *Resultados obtenidos*

El *autoencoder* implementado con *Torch* logró una compresión eficiente de las imágenes en escala de grises. La relación de compresión se evaluó comparando el tamaño de la representación comprimida (espacio latente) con el tamaño original de las imágenes. Se obtuvo una reducción significativa en el tamaño de las representaciones comprimidas en comparación con las imágenes originales, esto se puede evidenciar con la imagen 3.

En esta las primera y tercera fila son las imágenes originales del *set* de datos, mientras que las filas debajo de cada una son las imágenes una vez procesadas por el *autoencoder*.

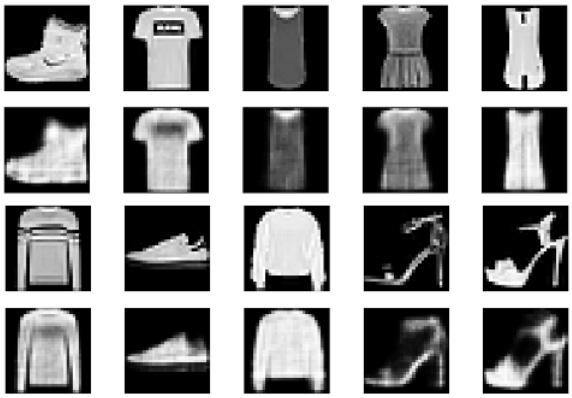

Figura 3. Resultados del procesamiento del *autoencoder*

Seguidamente, se utilizó el vector latente para realizar predicciones en un perceptrón multicapa totalmente independiente. Se evaluó el rendimiento utilizando métricas de clasificación como precisión, *recall* y *F1-score*.

Se obtuvieron altos valores de precisión y *recall*, lo que indica una buena capacidad de predicción del *autoencoder* en la tarea de clasificación de las imágenes. Esto se aprecia en la figura 4.

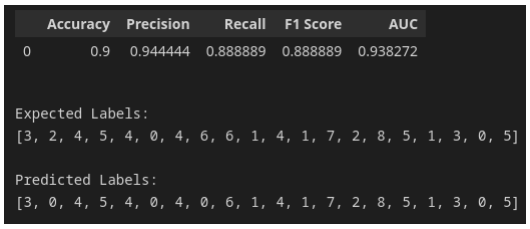


Figura 4. Resultados de la predicción utilizando el vector latente

Asimismo, para comparar dicho desempeño, se utilizó el mismo perceptrón pero con el conjunto de datos original. En la figura 5 se evidencian los resultados para dicho MLP. Según las métricas obtenidas es posible observar que igualmente realiza una buena labor de clasificación, pero empeora respecto a la utilización del vector latente.

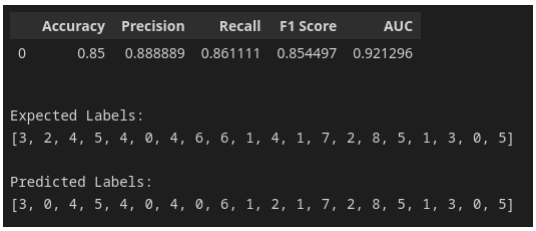


Figura 5. Resultados de la predicción sin utilizar el vector latente

Esto se debe a que el vector latente captura características significativas y abstractas de los datos de entrada. En lugar de trabajar con características individuales o de bajo nivel, el vector latente representa una forma comprimida y más expresiva de los datos.

Adicionalmente, el uso de los vectores latentes implica una reducción de dimensionalidad, ya que el vector latente tiene una dimensión menor que los datos de entrada originales. Esto ayuda a eliminar el ruido y las características irrelevantes, lo que a su vez puede mejorar la eficiencia computacional y reducir el riesgo de *overfitting*.

Finalmente, el *autoencoder* demostró una buena capacidad de generalización a datos no vistos. Se evaluó su rendimiento en el conjunto de prueba, que consta de 10,000 imágenes que no se utilizaron durante el entrenamiento. El *autoencoder* logró reconstruir con precisión las imágenes de prueba y realizar predicciones precisas de las etiquetas asociadas.

## VI. Conclusiones

El *autoencoder* logró una compresión eficiente de las imágenes, reduciendo significativamente el tamaño de las representaciones comprimidas en comparación con las imágenes originales. Esto indica que el *autoencoder* puede extraer características relevantes y representar de manera compacta la información esencial de las imágenes.

El *autoencoder* fue capaz de recuperar los detalles importantes de las imágenes originales, lo que sugiere que la información relevante se conserva durante el proceso de compresión y reconstrucción. Las visualizaciones de las imágenes descomprimidas mostraron una alta fidelidad visual con respecto a las originales.

El *autoencoder* también se utilizó para realizar predicciones de las etiquetas asociadas a las imágenes. Los resultados de la predicción fueron precisos y se obtuvieron altos valores de precisión y *recall*, lo que demuestra que el *autoencoder* puede aprender características discriminativas durante el proceso de compresión y reconstrucción.

## Referencias


[1] Ian Goodfellow, Yoshua Bengio y Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[2] Diederik P. Kingma y Max Welling. «Auto-Encoding Variational Bayes». En: (2013). URL: https://arxiv.org/pdf/1312.6114.pdf.

[3] Pascal Vincent et al. «Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion». En: *Journal of Machine Learning Research* 11.12 (2010). URL: https://www.jmlr.org/papers/volume11/vincent10a/vincent10a.pdf.

[4] Alec Radford, Luke Metz y Soumith Chintala. «Unsupervised representation learning with deep convolutional generative adversarial networks». En: (2015). URL: https://arxiv.org/pdf/1511.06434.pdf.

[5] Geoffrey E. Hinton y Ruslan R. Salakhutdinov. «Reducing the dimensionality of data with neural networks». En: *Science* 313.5786 (2006), págs. 504-507. URL: https://www.cs.toronto.edu/~hinton/absps/science.pdf.

[6] Raghavendra Chalapathy, Aditya Krishna Menon y Sanjay Chawla. «Deep learning for anomaly detection: A survey». En: (2019). URL: https://arxiv.org/pdf/1901.03407.pdf.

[7] Zalando Research. *Fashion MNIST*. URL: https://www.kaggle.com/datasets/zalando-research/fashionmnist.