Aplicación de Autoencoder para la Reconstrucción de Imágenes

Alexander Oviedo Fadul

Corporación Universitaria Minuto de Dios – UNIMINUTO

Especialización en Inteligencia Artificial y Deep Learning

NRC-198: Deep Learning

Profesor: Juan Carlos Valencia

27 de abril de 2026

Tabla de Contenido

1. Introducción

El presente documento constituye el entregable de la Semana 7 del módulo de Deep Learning. Su propósito es responder, con solidez teórica y experimentación práctica, las dos preguntas orientadoras formuladas en la actividad: (a) ¿Cómo trabaja un autoencoder? y (b) ¿Cómo se utiliza un autoencoder para la reconstrucción de imágenes con Keras y TensorFlow? El análisis se apoya en la bibliografía asignada —Bonaccorso (2018) y Fandango (2018)—, así como en los conceptos desarrollados durante la Open Class de la semana.

Desde mi experiencia profesional en la Rama Judicial de Colombia, los autoencoders representan una tecnología prometedora para tareas como la limpieza de documentos escaneados deteriorados, la detección de anomalías en flujos de datos masivos y la compresión de representaciones complejas. Ese contexto motiva un interés particular en comprender su funcionamiento a fondo, más allá del ejercicio puramente académico.

2. ¿Cómo Trabaja un Autoencoder?

2.1. Arquitectura General

Un autoencoder es una red neuronal de aprendizaje no supervisado cuyo objetivo es aprender una representación comprimida de los datos de entrada para luego reconstruirlos con la mayor fidelidad posible. Su arquitectura se compone de tres bloques funcionales: el codificador (encoder), el cuello de botella (bottleneck o espacio latente) y el decodificador (decoder). Como señala Bonaccorso (2018), el encoder reduce progresivamente la dimensionalidad de la entrada mediante capas sucesivas, forzando a la red a conservar únicamente la información más representativa.

El espacio latente constituye la capa central de la arquitectura. En la sesión de clase, el profesor Valencia lo describió como 'la imagen comprimida', un vector de dimensión reducida que captura la esencia de la señal de entrada. El decoder, por su parte, toma esta representación compacta y expande la información de vuelta hacia la dimensionalidad original, intentando reconstruir el dato de entrada con el menor error posible.

2.2. Entrenamiento No Supervisado

A diferencia de las redes supervisadas que requieren etiquetas explícitas, los autoencoders utilizan los propios datos de entrada como objetivo de salida. Es decir, la función de pérdida mide la diferencia entre la entrada original (x) y la salida reconstruida ($\hat{x}$). Tal como se discutió en clase, esta propiedad los convierte en herramientas particularmente útiles cuando no se dispone de conjuntos etiquetados, una situación frecuente en dominios como la administración de justicia donde los datos son abundantes pero las categorías formales son costosas de producir.

La función de pérdida más habitual para imágenes normalizadas en el rango [0, 1] es la entropía cruzada binaria (binary crossentropy). El optimizador Adam, respaldado por la experimentación del curso y la bibliografía de Fandango (2018), ha demostrado convergencia estable y eficiente para este tipo de arquitecturas.

2.3. Funciones de Activación

Las funciones de activación desempeñan un rol crítico. En las capas intermedias del encoder se emplea preferentemente ReLU (Rectified Linear Unit), que introduce no linealidad y mitiga el problema del gradiente evanescente. Para la capa final del decoder, cuando los valores de píxeles están normalizados entre 0 y 1, se utiliza la función sigmoide, que garantiza salidas

acotadas en ese intervalo. Esta combinación —ReLU interna, sigmoide final— fue explícitamente recomendada en el material de clase y confirmada en el notebook de referencia proporcionado por Keras (Valdarrama, 2021).

3. Reconstrucción de Imágenes con Keras y TensorFlow

3.1. Preparación de Datos (MNIST)

El conjunto de datos MNIST, compuesto por 60 000 imágenes de entrenamiento y 10 000 de prueba de dígitos manuscritos (28 × 28 píxeles en escala de grises), constituye el benchmark estándar para este tipo de ejercicios. Cada imagen se normaliza dividiendo sus valores de píxeles por 255, de modo que queden en el intervalo [0, 1]. Para el autoencoder denso, las imágenes se aplanan a vectores de 784 componentes; para el convolucional, se mantienen como tensores de 28 × 28 × 1.

3.2. Autoencoder Denso

El primer modelo implementado en el notebook del curso consiste en un autoencoder denso (fully connected) con una capa de entrada de 784 neuronas, una capa oculta de compresión con 32 neuronas (activación ReLU) y una capa de salida de 784 neuronas (activación sigmoide). El modelo se compila con el optimizador Adam y la función de pérdida binary_crossentropy. Después de 10 épocas de entrenamiento, el autoencoder reconstruye los dígitos con pérdidas cercanas a 0.09, produciendo reconstrucciones reconocibles aunque con cierto suavizado en los trazos, lo que evidencia la pérdida de detalle inherente a la compresión de 784 a 32 dimensiones.

3.3. Autoencoder Convolucional para Eliminación de Ruido

El segundo y más avanzado modelo utiliza capas convolucionales (Conv2D) en el encoder con filtros de 32 canales y pooling, seguidas de capas Conv2DTranspose en el decoder para recuperar la resolución original. A diferencia del modelo denso, este autoencoder preserva la estructura espacial bidimensional de las imágenes, lo que se traduce en reconstrucciones de mayor calidad.

Para demostrar la capacidad de eliminación de ruido, se añade ruido gaussiano con un factor de 0.4 a las imágenes de entrenamiento y prueba. El modelo se entrena con las imágenes ruidosas como entrada y las imágenes limpias como objetivo, de modo que aprende a filtrar el ruido y recuperar la señal subyacente. Tras 100 épocas de entrenamiento, las reconstrucciones eliminan exitosamente gran parte del ruido, validando la eficacia de los autoencoders convolucionales en tareas de denoising.

4. Experimentación y Resultados

En el notebook adjunto se implementaron las modificaciones solicitadas en la actividad: variación del número de dígitos mostrados (n = 5), cambio de optimizador (se probó SGD con learning rate 0.001 y comparó contra Adam), modificación de epochs y batch_size, incorporación de una capa oculta adicional con 256 neuronas, aplicación de Dropout al 20 %, y cambio de la función de pérdida a mean_squared_error. Los resultados se visualizan comparando imágenes originales contra reconstruidas, confirmando que Adam con binary_crossentropy y Dropout regularizado ofrece la mejor relación pérdida/calidad.

Desde una óptica profesional, estos experimentos revelan que la correcta calibración de hiperparámetros no es un ejercicio trivial sino un proceso empírico informado por la teoría. La experiencia adquirida en semanas anteriores con KerasTuner resulta directamente transferible a

este contexto, reforzando la importancia de un enfoque sistemático para la optimización de modelos de Deep Learning.

5. Conclusiones

Los autoencoders constituyen una arquitectura versátil y conceptualmente elegante dentro del repertorio del Deep Learning. Su capacidad para aprender representaciones comprimidas de forma no supervisada los posiciona como herramientas valiosas en escenarios donde los datos etiquetados son escasos o costosos de producir. La implementación práctica con Keras y TensorFlow demuestra que, con pocas líneas de código y la configuración adecuada de hiperparámetros, es posible construir modelos capaces de reconstruir imágenes e incluso eliminar ruido con resultados notables.

En el contexto de la transformación digital judicial, imagino aplicaciones concretas como la restauración de folios escaneados con deterioro, la compresión eficiente de archivos de imagen en expedientes digitales y la detección de anomalías en flujos documentales. La Semana 7 reafirma que el Deep Learning no es únicamente un campo de investigación abstracta, sino una caja de herramientas con potencial transformador directo en la administración pública.

6. Enlace al Notebook (Google Colab)

Enlace Jupyter Notebook (Google Colab):

https://colab.research.google.com/drive/1ngdVY9ZDAgRidzUqRBsyEpToP1phKShS

7. Referencias

Bonaccorso, G. (2018). Mastering Machine Learning Algorithms. Packt Publishing.

Fandango, A. (2018). Mastering TensorFlow 1.x. Packt Publishing.

Valdarrama, S. L. (2021). Convolutional autoencoder for image denoising. Keras. https://keras.io/examples/vision/autoencoder/

Valencia, J. C. (2026). Semana 7. Aplicación de autoencoder para la reconstrucción de imágenes [Transcripción y Recursos del Curso]. Corporación Universitaria Minuto de Dios – UNIMINUTO.