Inteligencia Artificial
Apuntes Semana 11, Clase #1

Luis Fernando Benavides Villegas
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
lubenavides@estudiantec.cr

***Abstract*—Este documento recopila los apuntes de la clase del martes 14 de octubre de 2025 para el curso de Inteligencia Artificial. Se repasan conceptos como los fundamentos de las redes neuronales convolucionales (CNN), abarcando el uso de filtros, stride, padding y pooling para la extracción de características en imágenes. Se analizan arquitecturas representativas como LeNet, AlexNet, GoogleNet, VGG, ResNet y DenseNet, destacando su evolución y aportes al aprendizaje profundo. Además, se introducen los conceptos de embeddings y visualización de activaciones para interpretar el comportamiento de los modelos, junto con el estudio de los autoencoders y su aplicación en reconstrucción de imágenes, reducción de dimensionalidad, detección de anomalías y generación de datos.**

***Index Terms*—Inteligencia Artificial, Redes Neuronales Convolucionales, Pooling, Embeddings, Visualización, Autoencoder, Deep Learning.**

I. Repaso de la Clase Anterior

A. Convoluciones y Filtros

Una convolución consiste en aplicar un filtro (kernel) sobre una imagen para extraer información relevante. El filtro es una matriz de números que se entrena junto con la red. Al desplazarse por la imagen, calcula un valor por cada posición, generando una nueva imagen llamada *mapa de características* (*feature map* o *activation map*).

1) Filtro Gaussiano: Produce una imagen con un efecto de desenfoque (blur), eliminando el ruido y dejando solo la parte del contorno.

2) Redes Neuronales: En redes convencionales, todos los píxeles estan conectados a todas las neuronas de la siguiente capa. En las convoluciones, solo una porción de los píxeles está conectada, observando solo una parte específica de la imagen.

3) Campo Receptivo: Es la región de la imagen que una neurona observa para generar su salida. Depende tanto del tamaño de la entrada como del filtro aplicado. Por ejemplo, si la imagen de entrada es de $32 \times 32 \times 3$, la red debe procesar los tres canales de color. Si el filtro tiene tamaño 5×5, entonces el campo receptivo resultante será un cubo de $5 \times 5 \times 3$, es decir, todas las neuronas que intervienen en esa región. Estos campos extraen características necesarias para el clasificador.

B. Parámetros de la Convolución

1) Stride: Define cuánto se deslizan los filtros sobre la imagen de entrada. Un *stride* mayor provoca menos posiciones de aplicación del filtro, por lo que reduce el tamaño del mapa de salida.

2) Padding: Agrega píxeles de relleno alrededor de la imagen de entrada para controlar el tamaño del mapa de salida y preservar las dimensiones espaciales. Un *padding* simétrico evita que la convolución reduzca el tamaño de la imagen:

$$p = \frac{k-1}{2},$$

donde k es el tamaño del filtro.

3) Dimensión de Salida: Dada una imagen de entrada y un kernel, el tamaño de salida se calcula como:

$$\frac{(m + 2P - K)}{S} + 1,$$

donde m es el tamaño de la entrada, K el tamaño del filtro, P el *padding* y S el *stride*.

C. Compartición de Pesos

En una red convolucional, los mismos pesos que se calcularon para una región específica se reutilizan en todas las demás posiciones donde el filtro se deslice. Por ejemplo, si un filtro aprende a detectar líneas verticales, esa misma configuración de pesos servirá para reconocerlas sin importar en qué parte de la imagen aparezcan. De esta manera, se reduce significativamente la cantidad de parámetros a entrenar y mejora la eficiencia del modelo.

En arquitecturas como *AlexNet*, permite que las primeras capas aprendan características generales como bordes y colores, mientras que las capas más profundas combinan esa información para reconocer formas y objetos más complejos.

D. Capa de Pooling

Después de aplicar las convoluciones, se utiliza una *capa de pooling* para reducir el tamaño espacial de la imagen y mantener solo la información más relevante. Esta operación toma bloques locales y realiza una operación estadística sobre ellos, como el máximo o el promedio, para resumir su contenido.

- **Max Pooling:** selecciona el valor máximo de cada bloque. Es el método más utilizado.
- **Average Pooling:** calcula el promedio de los valores de cada bloque.
- **L2 Pooling:** aplica una norma cuadrática sobre los valores.

El *pooling* reduce el ancho y el alto de la imagen, pero conserva la cantidad de canales, por lo que con entrada de tamaño $W \times H \times D$, el pooling reduce W y H, manteniendo D. Esto evita que el modelo crezca en cantidad de parámetros y mantiene la información esencial para las siguientes capas.

E. Capa Fully-Connected

Tras las etapas de convolución y pooling, la red produce un vector que resume las características más relevantes de la imagen. Este vector se conecta a una o varias *capas totalmente conectadas*. Cada neurona de estas capas está conectada con todas las salidas anteriores, permitiendo combinar las características extraídas para realizar la clasificación final.

El perceptrón multicapa (MLP) se encarga de transformar este vector en una predicción, como la probabilidad de pertenencia a una clase específica.

F. Arquitecturas Convolucionales

1) Estructura General: Una arquitectura convolucional se compone de bloques repetidos de:

$$\text{Convolución} \rightarrow \text{Activación} \rightarrow \text{Pooling}$$

Estos bloques se repiten varias veces para extraer información progresivamente más abstracta. Posteriormente, el resultado se aplana (*flatten*) y se conecta a una o más capas *fully connected* para la clasificación.

Se recomienda el uso de filtros pequeños (por ejemplo, 3×3 o 5×5) ya que permiten:

- Reducir la cantidad de parámetros a aprender.
- Capturar relaciones no lineales al encadenar múltiples capas.

Filtros grandes (7×7 o más) capturan más información en una sola capa, pero aumentan excesivamente el número de parámetros y reducen la no linealidad.

2) Reglas Prácticas:

- Es preferible que las dimensiones de las imágenes sean divisibles entre 2 para facilitar las reducciones con *max pooling*.
- En general, se utiliza *stride* de 2 y *padding* de 1 para mantener dimensiones manejables.
- El *pooling* de 2×2 es el más común, reduciendo la imagen a la mitad en cada dimensión.

3) Principales Arquitecturas:

a) LeNet-5: Propuesta por *Yann LeCun* en 1998, fue una de las primeras redes convolucionales aplicadas al reconocimiento de dígitos escritos a mano [1]. Su estructura incluye dos capas convolucionales, dos de *pooling* y una totalmente conectada, estableciendo la base para las redes modernas de visión por computadora.

b) AlexNet: Desarrollada por *Krizhevsky, Sutskever* y *Hinton* en el 2012, marcó el inicio del *Deep Learning* moderno. Procesa imágenes de 224×224 con filtros grandes (11×11, 5×5, 3×3), emplea activaciones *ReLU*, *dropout* y entrenamiento distribuido en múltiples GPUs, logrando un salto significativo en precisión sobre el conjunto ImageNet.

c) ZFNet: Creada en base a AlexNet, ajusta el tamaño de los filtros y la profundidad para estudiar cómo cada capa transforma la información. Introdujo técnicas para visualizar activaciones intermedias, ayudando a comprender y depurar el comportamiento interno de las CNN.

d) GoogleNet (Inception): Presentada por Google en 2014, redujo de 60 a 4 millones de parámetros mediante los módulos *Inception*, que combinan convoluciones de distintos tamaños (1×1, 3×3, 5×5) y *max pooling* en paralelo. En la etapa final, un *average pooling* global transforma el tensor de $7 \times 7 \times 1024$ en un vector $1 \times 1 \times 1024$, reemplazando las capas densas y mejorando la eficiencia [2].

e) VGG16: Simplifica el diseño utilizando solo filtros pequeños de 3×3 y bloques repetidos de convolución y *pooling*. Aumenta la profundidad hasta 16 o 19 capas, mostrando que más capas con filtros simples mejoran el rendimiento general.

f) ResNet: Introduce las *conexiones residuales*, que permiten que la información fluya entre capas no consecutivas. Estas conexiones evitan el desvanecimiento del gradiente y posibilitan entrenar redes extremadamente profundas de forma estable.

g) DenseNet: Conecta cada capa con todas las anteriores dentro de un bloque, promoviendo la reutilización de características y reduciendo la redundancia. Esta estructura densa mejora la propagación del gradiente, optimiza la eficiencia del modelo y mantiene un número reducido de parámetros.

II. Problemas con las Redes Neuronales Convolucionales

A pesar de su alto desempeño, las redes convolucionales se comportan como una "caja negra", ya que resulta difícil comprender qué tipo de información están utilizando para tomar sus decisiones. Las representaciones internas que generan son altamente abstractas, lo que plantea un reto importante de interpretabilidad.

Uno de los principales desafíos actuales es entender qué es lo que realmente afecta a la red durante el proceso de clasificación. Las capas internas aprenden características complejas que no siempre son comprensibles para los humanos. Este problema de explicabilidad motiva el uso de técnicas de visualización que permitan analizar la respuesta de las neuronas ante diferentes estímulos visuales.

A. Visualización y Análisis de Activaciones

Una forma práctica de estudiar el comportamiento interno de las CNN es observar los *feature maps* generados por cada capa.

En estos mapas se puede identificar qué regiones de la imagen activan ciertas neuronas y, por lo tanto, cuáles son los elementos visuales que el modelo considera relevantes. En las primeras capas, las activaciones suelen asemejarse todavía a la imagen original, pero conforme se profundiza, las representaciones se vuelven cada vez más abstractas y difíciles de interpretar.

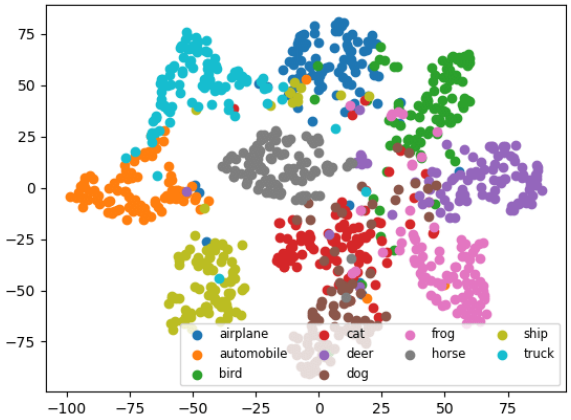

Fig. 1. Representación de clases con t-SNE.

Además, la inspección de los filtros aprendidos ayuda a verificar qué patrones está capturando la red. Los filtros iniciales tienden a mostrar texturas, bordes o colores, mientras que los últimos codifican composiciones más complejas. Este tipo de análisis permite detectar si la red está aprendiendo características significativas o simplemente ruido del conjunto de entrenamiento.

B. Reducción de Dimensionalidad

Para comprender las redes, se puede hacer el estudio de sus *embeddings*. Al final de la red, cada imagen puede representarse mediante un vector numérico que resume su información semántica. Imágenes similares quedan próximas entre sí en este espacio vectorial, mientras que las de distintas clases se separan claramente. Estas representaciones pueden visualizarse mediante algoritmos de reducción de dimensionalidad, como:

- **t-SNE:** proyecta los vectores a dos o tres dimensiones, preservando la estructura de las distancias originales, como se puede ver en la Fig. 1.
- **PCA:** alternativa más simple, aunque menos efectiva para relaciones no lineales.

Cuando la separación entre clases es clara en el espacio reducido, se considera que el modelo ha aprendido una representación adecuada. Por el contrario, si las clases aparecen mezcladas, indica que la red no ha logrado distinguir correctamente las características de cada una.

C. Mapas de Activación

Se pueden generar *heatmaps* o mapas de activación que destacan las zonas específicas de una imagen que influyen más en la decisión del modelo. Estos mapas son útiles para verificar si la red está enfocándose en las regiones correctas del objeto. Por ejemplo, estos métodos permiten justificar predicciones, como localizar una fractura o anomalía en una radiografía, proporcionando transparencia al proceso de decisión del modelo.

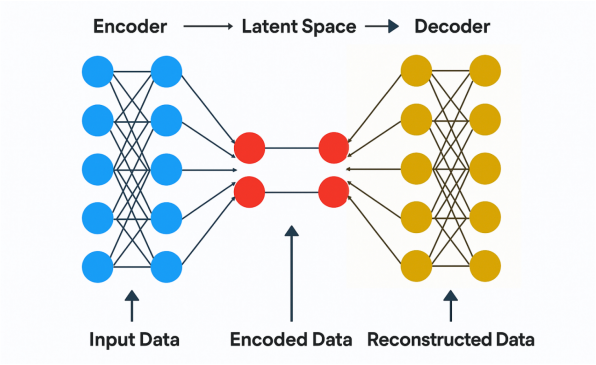

Fig. 2. Diagrama del funcionamiento de un Autoencoder

III. Autoencoders

Es una red neuronal diseñada para aprender una representación comprimida de sus datos de entrada. A diferencia de las redes supervisadas, no necesita etiquetas externas, ya que su objetivo es reconstruir la entrada en la salida. Durante el entrenamiento, el modelo aprende a capturar los patrones más relevantes de los datos, filtrando el ruido y conservando la información esencial.

La estructura básica se compone de tres partes (Fig. 2):

$$\text{Encoder} \rightarrow \text{Espacio Latente} \rightarrow \text{Decoder}$$

El aprendizaje del autoencoder consiste en minimizar el *error de reconstrucción* entre la entrada original y la salida reconstruida. Aunque no haya etiquetas externas, el entrenamiento es parcialmente supervisado, ya que la salida se compara directamente con la entrada.

A. Encoder

Consiste en una serie de bloques convolucionales seguidos de operaciones de *pooling*, con el objetivo de extraer las características más relevantes de la entrada y comprimir la información a través de un proceso de reducción espacial o *downsampling*. Cada bloque convolucional aprende distintos niveles de representación, pasando de detalles simples como bordes y texturas a rasgos más abstractos. De esta manera, el encoder transforma los datos originales en una versión más compacta, conservando únicamente la información esencial para la reconstrucción posterior.

B. Espacio Latente o Cuello de Botella

En esta etapa se almacena la información esencial en un vector de baja dimensionalidad, conocido como *espacio latente*. Este vector contiene la codificación interna de la entrada, capturando únicamente los rasgos más significativos. Debido a su tamaño limitado, restringe el flujo de información hacia el decoder, lo que obliga al modelo a conservar solo lo más relevante para lograr una reconstrucción efectiva. En este espacio, muestras similares tienden a ubicarse próximas

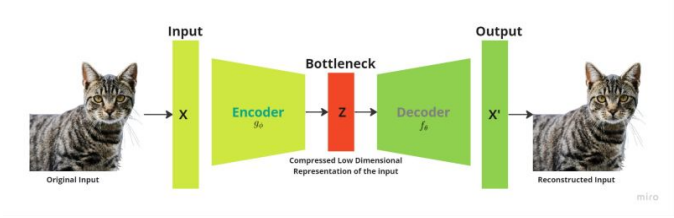

Fig. 3. Ejemplo de Super-Resolución con Autoencoder.

entre sí, formando agrupamientos que reflejan la estructura semántica de los datos.

C. Decoder y Reconstrucción

A partir del vector del espacio latente, utiliza capas de *upsampling* o *convoluciones transpuestas* para expandir progresivamente la representación comprimida hasta recuperar la forma original. Durante este proceso, el modelo aprende a reconstruir los detalles perdidos, generando una salida que se asemeje lo más posible a la entrada inicial.

D. Aplicaciones de los Autoencoders

- **Reducción de dimensionalidad:** obtener representaciones más compactas que las de PCA.
- **Detección de anomalías:** los ejemplos normales se reconstruyen bien, mientras que los atípicos muestran un error de reconstrucción elevado. Se puede fijar un umbral para decidir cuándo un dato es anómalo.
- **Eliminación de ruido:** aprender a reconstruir una imagen limpia a partir de una ruidosa.
- **Edición y generación de imágenes:** al modificar el vector latente se pueden crear variantes o nuevas imágenes, por ejemplo, para comprimirlas.
- **Super-Resolución:** generar versiones de alta resolución a partir de imágenes pequeñas (Fig. 3).

E. Hiperparámetros Relevantes

- **Tamaño del vector latente:** define la cantidad de información que el modelo puede retener en el espacio comprimido. Un vector más pequeño produce un modelo más eficiente en cómputo, pero con menor capacidad para capturar detalles de la imagen. En cambio, un vector más grande permite representar más características, aunque incrementa el costo de entrenamiento y de procesamiento.
- **Número de capas:** tanto el encoder como el decoder pueden variar en profundidad. Un mayor número de capas permite modelar relaciones más complejas, pero también hace el entrenamiento más pesado y sensible al ajuste de parámetros.
- **Función de pérdida:** para tareas de reconstrucción de imágenes se utiliza comúnmente el *Mean Squared Error (MSE)*. Esta función compara cada píxel de la imagen original con el de la reconstrucción, midiendo su diferencia. Un error cercano a cero indica que el modelo ha logrado reproducir correctamente la entrada.

IV. Conclusiones

Las redes convolucionales permiten extraer automáticamente características jerárquicas de las imágenes, impulsando el desarrollo de arquitecturas cada vez más profundas y eficientes. A pesar de su potencia, siguen siendo poco interpretables, por lo que se recurre a técnicas de visualización y análisis de activaciones. Finalmente, los *autoencoders* amplían estos conceptos al aprendizaje no supervisado, permitiendo la compresión, reconstrucción y generación de datos a partir de representaciones latentes.

Referencias

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] C. Szegedy et al., "Going deeper with convolutions," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015.