Apuntes IA Clase 14/10/2025

Juan Jiménez Valverde
Escuela de Ingeniería en Computación
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
juand0908@estudiantec.cr

***Abstract*—Este documento resume los conceptos clave vistos en la clase de Inteligencia Artificial sobre redes neuronales convolucionales (CNN) y autoencoders. Se abordan temas fundamentales como los filtros, campos receptivos, *stride*, *padding* y las capas de *pooling*, así como las arquitecturas clásicas de CNN, entre ellas LeNet, AlexNet, ZFNet, GoogleNet, VGG16, ResNet y DenseNet. Además, se presentan consideraciones prácticas para el análisis de modelos de aprendizaje profundo, como la visualización de activaciones, los *embeddings* de características y la estructura de los autoencoders, incluyendo sus aplicaciones en reducción de dimensionalidad, detección de anomalías y procesamiento de imágenes. El objetivo es ofrecer una visión general clara y concisa, útil tanto para el estudio teórico como para la aplicación práctica.**

***Index Terms*—Redes Neuronales Convolucionales, Autoencoders, Visualización de Activaciones, Embeddings, Pooling, Arquitecturas de Aprendizaje Profundo**

I. Introducción

Las Redes Neuronales Convolucionales (CNN) se han convertido en un pilar fundamental de la visión por computadora moderna, ya que permiten extraer automáticamente características jerárquicas a partir de imágenes. Comprender sus mecanismos internos, incluyendo los filtros, los campos receptivos, el *stride*, el *padding* y las operaciones de *pooling*, resulta esencial para diseñar arquitecturas eficientes e interpretar el comportamiento de los modelos.

Por su parte, los **autoencoders** complementan el uso de las CNN al aprender representaciones compactas de los datos sin requerir etiquetas, lo que los hace ideales para tareas de aprendizaje no supervisado como la reducción de dimensionalidad, la detección de anomalías y la reconstrucción de imágenes.

Este documento sintetiza los principales conceptos y arquitecturas revisados en clase, sirviendo como material de referencia para la comprensión y aplicación práctica de estos modelos en inteligencia artificial.

II. Repaso de la clase anterior

A. Filtros o Kernels

Son matrices (por ejemplo, de 3×3 o 5×5) que se deslizan sobre la imagen para aplicar convoluciones. El *Gaussian Kernel* se utiliza para suavizar la imagen y eliminar ruido.

B. Campo Receptivo

Es la región local de la imagen a la que una neurona está conectada. Por ejemplo, para una entrada de $32 \times 32 \times 3$ y un campo receptivo de 5×5, cada neurona tendrá $5 \times 5 \times 3 = 75$ pesos.

C. Stride, Padding y Cálculo de Dimensiones

- **Stride:** Indica cuántos pasos da el filtro al desplazarse sobre la imagen. Un *stride* mayor reduce el tamaño de la salida.
- **Padding:** Agrega píxeles (usualmente ceros) alrededor de la imagen de entrada para controlar el tamaño de salida y preservar dimensiones. El padding simétrico típico se calcula como:

$$p = \frac{(k-1)}{2}$$

  donde k es el tamaño del kernel.

El tamaño de salida se calcula con:

$$\text{Dimensión de salida} = \frac{(m - k + 2p)}{s} + 1$$

donde: m es el tamaño de la entrada, k el tamaño del kernel, p el padding aplicado y s el stride.

D. Pesos y Arquitectura de AlexNet

En redes convolucionales se utilizan **pesos compartidos**, lo que reduce drásticamente el número de parámetros, ya que el mismo conjunto de filtros se aplica en todas las posiciones espaciales. En la arquitectura de AlexNet, por ejemplo, se emplean 96 filtros en la primera capa convolucional, permitiendo extraer múltiples características visuales de forma eficiente.

E. Pooling Layer

Después de las convoluciones, se aplica la capa de *pooling*, que resume la información espacial (alto y ancho) sin alterar la cantidad de canales. Existen dos tipos principales:

- **Max Pooling:** conserva el valor máximo de cada región.
- **Average Pooling:** calcula el promedio de los valores en la región.

Dada una entrada de tamaño $W \times H \times D$, el *pooling* reduce W y H, manteniendo D.

F. Fully-Connected Layer

Finalmente, las características extraídas se transforman en un único vector, conectando todas las neuronas entre sí. Esta capa permite realizar la clasificación final del modelo.

G. Arquitecturas Convolucionales

Una red convolucional combina secuencias de **convolución → activación (ReLU) → pooling**. Este patrón se repite varias veces para extraer información progresivamente más abstracta de la imagen. Generalmente, se prefieren filtros pequeños (como 3×3) para capturar detalles locales de forma más eficiente.

El *convolutional stack* se forma al aplicar múltiples capas de convolución consecutivas. Por ejemplo, en una imagen de 5×5, un filtro 3×3 puede desplazarse para generar una salida de 3×3.

Regla práctica: las dimensiones de las imágenes deben ser divisibles entre 2, lo cual facilita la reducción progresiva mediante pooling.

H. Principales Arquitecturas

1) LeNet: Diseñada por Yann LeCun et al. (1998), fue una de las primeras redes convolucionales exitosas. Cuenta con 5 capas: dos convolucionales, dos de *pooling* y una totalmente conectada [1]. Su estructura sirvió de base para las redes modernas.

2) AlexNet: Propuesta por Krizhevsky, Sutskever y Hinton (2012), marcó un hito en la visión por computadora. Procesa imágenes de 224×224 con filtros grandes (11×11, 5×5, 3×3) y cinco capas convolucionales. Popularizó el uso de *ReLU*, *dropout* y la utilización de múltiples GPUs para el entrenamiento [2].

3) ZFNet: Basada en AlexNet, reduce la profundidad y tamaño de los filtros para analizar cómo afectan las capas a la representación interna. Sirvió como experimento para visualizar activaciones intermedias y optimizar arquitecturas.

4) GoogleNet (Inception): Reduciendo los más de 60 millones de parámetros de AlexNet a unos 4 millones, GoogleNet introdujo los módulos *Inception* [3]. Cada módulo combina convoluciones de diferentes tamaños (1×1, 3×3, 5×5) junto con *max pooling*, concatenando sus resultados. Al final, la salida ($7 \times 7 \times 1024$) se aplana y se pasa a un *average pooling* de $1 \times 1 \times 1024$.

5) VGG16: Caracterizada por su simplicidad, utiliza únicamente filtros de 3×3 y aumenta la profundidad hasta 16 capas. Esta arquitectura demostró que aumentar la profundidad mejora el rendimiento si se mantienen filtros pequeños y consistentes.

6) ResNet: Introduce las **conexiones residuales**, que permiten el paso de información entre capas no adyacentes. Esto evita la degradación del gradiente en redes muy profundas y mejora la capacidad de entrenamiento.

7) DenseNet: Conecta cada capa con todas las anteriores, favoreciendo la reutilización de características y reduciendo la cantidad de parámetros necesarios. Este enfoque mejora la eficiencia y el flujo de información a lo largo de la red.

III. Materia de clase

A. Problemas en las Redes Neuronales Convolucionales

1) Explicabilidad del Modelo: Uno de los principales desafíos actuales es la falta de interpretabilidad en las redes profundas. Los *features* aprendidos por las capas internas suelen ser difíciles de entender por los humanos, lo que complica saber qué está "viendo" realmente el modelo.

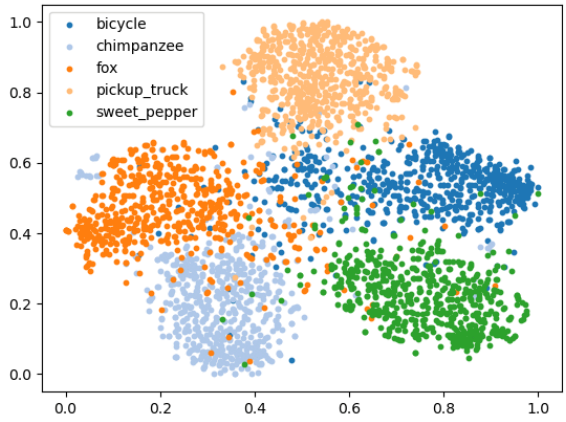

Fig. 1. Representación de embeddings mediante t-SNE.

2) Visualización y Análisis de Activaciones: Una forma de entender mejor el funcionamiento interno es observar:

- **Visualización de activaciones:** muestra qué regiones de la imagen activan ciertas neuronas.
- **Visualización de filtros:** permite observar los pesos de los kernels. En las primeras capas, estos muestran patrones reconocibles (bordes, colores, texturas), mientras que en capas profundas se vuelven más abstractos.

Estos métodos ayudan a detectar si el modelo está aprendiendo características relevantes o solo ruido.

3) Embeddings y Reducción de Dimensionalidad: Las redes pueden transformar imágenes en representaciones vectoriales llamadas *embeddings*. Estas representaciones condensan la información relevante de una imagen, permitiendo separar clases en el espacio de características. Al reducir la dimensionalidad (manteniendo las distancias relativas), podemos visualizar las relaciones entre clases.

Una técnica común para ello es **t-SNE**, que proyecta estos vectores a dos dimensiones preservando la estructura del espacio original (Fig. 1).

4) Mapas de Activación: Además de las visualizaciones de filtros, es posible generar *mapas de activación* o *heatmaps* que muestran qué regiones específicas de la imagen influyen más en la decisión del modelo. Estas técnicas son útiles, por ejemplo, en aplicaciones médicas para resaltar fracturas o anomalías en radiografías.

B. Autoencoders

Aunque utilizan arquitecturas similares a las redes convolucionales, los **autoencoders** trabajan sin etiquetas explícitas, por lo que se consideran métodos no supervisados. Su objetivo es reconstruir la entrada original, aprendiendo una representación interna comprimida.

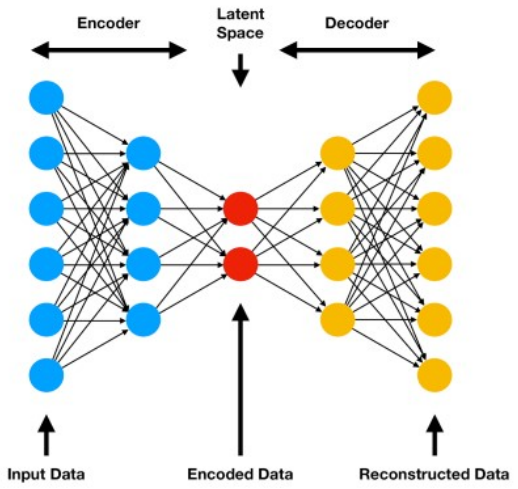

Fig. 2. Estructura básica de un Autoencoder.

El proceso consta de tres partes, como se muestra en la Fig. 2:

1) **Encoder:** reduce la imagen a un vector compacto.
2) **Espacio latente:** contiene la representación esencial o codificada de la entrada.
3) **Decoder:** reconstruye la imagen original a partir del vector latente.

1) Tareas Comunes de un Autoencoder:

- **Reducción de dimensionalidad:** genera una representación más compacta y poderosa que PCA, conservando la información esencial.
- **Detección de anomalías:** se entrena para reconstruir datos de una tarea tomando en cuenta únicamente ejemplos positivos o normales. Por ejemplo:
  - Transferencias bancarias correctas.
  - Audio o imágenes de alta fidelidad sin defectos.

  El modelo aprende la representación latente de estos casos y, al presentarle ejemplos anómalos, su reconstrucción falla, evidenciando la anomalía.
- **Procesamiento de imágenes (Fig. 3):** permite tareas como compresión, eliminación de ruido o incluso *super resolución*, es decir, generar imágenes de alta resolución a partir de versiones borrosas o pequeñas.

Estos principios sientan las bases de los algoritmos generativos modernos, donde el modelo aprende a reconstruir o crear contenido visual de forma autónoma.

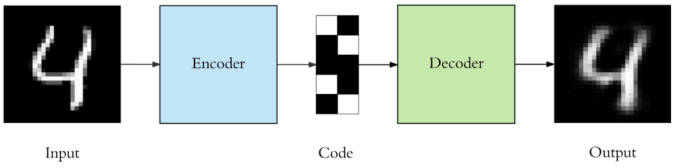

Fig. 3. Procesamiento de imágenes

IV. Partes del Autoencoder

Los autoencoders se componen de tres partes principales: el **encoder**, el **cuello de botella** y el **decoder**. Cada una cumple una función específica en el proceso de codificación y reconstrucción de los datos.

A. Encoder

El encoder está formado por un conjunto de bloques convolucionales seguidos de módulos de *pooling*. Su función principal es extraer las características más relevantes de la imagen de entrada y comprimir la información. La expectativa del encoder es aprender información importante de la entrada mediante un proceso de *downsampling*, reduciendo la dimensionalidad y conservando los rasgos esenciales.

B. Cuello de botella

El cuello de botella constituye la parte más importante y pequeña del modelo. Representa la información comprimida en un **espacio latente**, donde se encuentran codificadas las características más significativas. Esta capa restringe el flujo de información proveniente del encoder al decoder, limitando la cantidad de datos que pueden ser reconstruidos.

C. Decoder

El decoder está compuesto por una serie de convoluciones que realizan *upsampling* para reconstruir la imagen original a partir del vector latente. En *PyTorch*, esta tarea suele implementarse mediante capas `ConvTranspose2d`. El objetivo del decoder es generar una salida lo más fiel posible a la entrada original.

D. Hiperparámetros a considerar

El desempeño del autoencoder depende en gran medida de los hiperparámetros seleccionados, entre los que destacan:

- **Tamaño de la codificación (vector latente):** determina el nivel de compresión de los datos. Un tamaño menor implica mayor compresión, pero puede perderse información relevante.
- **Número de capas:** define la profundidad del encoder y del decoder. Un número mayor de capas genera un modelo más complejo y con mayor capacidad de representación, mientras que un número menor lo hace más rápido pero menos preciso.

V. Conclusiones

Durante la clase se destacaron los componentes esenciales y las arquitecturas principales de las redes neuronales convolucionales, explicando cómo las capas, filtros y operaciones de *pooling* trabajan en conjunto para extraer información relevante de las imágenes. La visualización de activaciones y *embeddings* permite comprender mejor el funcionamiento interno de los modelos profundos, mejorando su interpretabilidad y facilitando el diagnóstico de su desempeño.

Asimismo, los **autoencoders** se presentaron como herramientas potentes dentro del aprendizaje no supervisado, capaces de comprimir información, detectar anomalías y mejorar la calidad de las imágenes mediante su reconstrucción.

Dominar estos conceptos proporciona una base teórica y práctica sólida para el diseño, análisis y aplicación efectiva de modelos de aprendizaje profundo en distintos contextos.

REFERENCES

[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] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems (NIPS)*, pp. 1097–1105, 2012.

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