Maestría en Cómputo Estadístico (2025), 1–4
Curso - Análisis de Imágenes con Deep Learning

Tarea 2

Implementación de Arquitecturas Convolucionales para Clasificación de Imágenes

Gustavo Hernández Angeles[1]
[1]CIMAT, A.C., Unidad Monterrey, N.L., México

Abstract

This guide is for authors who are preparing papers for the *Publications of the Astronomical Society of Australia* journal using the LaTeX document preparation system and the CUP PAS style file.

1. Introducción

La clasificación de imágenes es una de las tareas fundamentales en el campo de la visión por computadora, cuyo propósito es asignar una etiqueta o categoría a una imagen digital basándose en su contenido visual. En las últimas décadas, este problema ha sido abordado con gran éxito mediante el uso de Redes Neuronales Convolucionales (CNNs). Estas arquitecturas han demostrado ser herramientas poderosas para la extracción automática de características jerárquicas, desde bordes simples hasta patrones complejos, permitiendo a los modelos aprender representaciones robustas de los datos visuales.

El objetivo principal de este reporte es evaluar y comparar el desempeño relativo de una selección de arquitecturas representativas en la historia del aprendizaje profundo. Específicamente, se analizan las arquitecturas LeNet5, AlexNet, VGG16, GoogleNet, MobileNetV1, MobileNetV2 y ResNet18. Para medir su eficacia y capacidad de generalización, se realizan experimentos de clasificación utilizando dos conjuntos de datos estándar en la literatura: MNIST, que consiste en dígitos manuscritos, y CIFAR10, que contiene imágenes de objetos y animales en 10 clases distintas. A través de estos experimentos, se busca comprender las fortalezas y limitaciones de cada modelo en función de su complejidad y precisión.

2. Arquitecturas Convolucionales

En esta sección se describen las arquitecturas convolucionales implementadas para la clasificación de imágenes. Cada arquitectura tiene características únicas que influyen en su desempeño y eficiencia.

2.1. LeNet-5

LeNet-5 es una arquitectura pionera en el campo de las redes neuronales convolucionales, propuesta por LeCun et al. (2002) para el reconocimiento de dígitos manuscritos. Esta red demostró la efectividad del aprendizaje basado en gradientes para tareas de visión por computadora.

Author for correspondence:
Cite this article:

La arquitectura (Figura 1) consta de siete capas entrenables (sin incluir la entrada): tres capas convolucionales (C1, C3, C5), dos capas de submuestreo (S2, S4) y dos capas totalmente conectadas (F6 y la capa de salida). Utiliza funciones de activación sigmoides o tangentes hiperbólicas en su versión original, y Average Pooling (como submuestreo) para reducir la dimensionalidad espacial. Aunque simple en comparación con modelos modernos, LeNet-5 estableció los principios fundamentales de las CNNs: campos receptivos locales, pesos compartidos y jerarquía espacial de características.

2.1.1. Detalles de la Implementación

La implementación realizada en este trabajo sigue la estructura general de LeNet-5 pero incorpora adaptaciones modernas para mejorar el entrenamiento. A diferencia del artículo original que empleaba funciones de activación sigmoides o tangentes hiperbólicas, nuestra implementación utiliza la función de activación Rectified Linear Unit (ReLU) después de cada capa convolucional y totalmente conectada (excepto la capa de salida), lo cual ayuda a mitigar el problema del desvanecimiento del gradiente y acelera la convergencia.

La red recibe como entrada imágenes en escala de grises de 32×32 píxeles. La primera capa convolucional aplica 6 filtros de tamaño 5×5, seguida de una capa de *Average Pooling* de 2×2. La segunda capa convolucional incrementa la profundidad a 16 filtros de 5×5, nuevamente seguida por *Average Pooling*. Finalmente, las características se aplanan en un vector de 400 dimensiones que alimenta a tres capas totalmente conectadas con 120, 84 y 10 neuronas respectivamente, correspondiendo esta última al número de clases del conjunto de datos.

2.2. AlexNet

AlexNet, propuesta por Krizhevsky et al. (2012), marcó un punto de inflexión en la visión por computadora al ganar la competencia ImageNet Large Scale Visual Recognition Challenge (ILSVRC) en 2012 con un margen significativo. La arquitectura original fue diseñada para clasificar imágenes de alta resolución (224×224) en 1000 clases. Consta de ocho capas aprendibles: cinco capas convolucionales y tres capas totalmente conectadas. Las primeras capas utilizan filtros grandes (e.g., 11×11) y strides agresivos

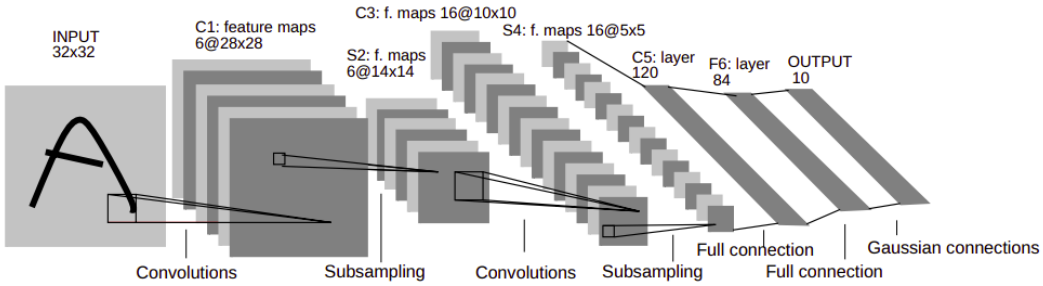

Figure 1. Arquitectura de la red LeNet-5 (LeCun et al. 2002).

para capturar características globales rápidamente. Además, introdujo innovaciones cruciales como el uso de funciones de activación ReLU para acelerar el entrenamiento y capas de Dropout para prevenir el sobreajuste en una red con 60 millones de parámetros.

2.2.1. Detalles de la Implementación

Nuestra implementación de AlexNet ha sido adaptada para procesar imágenes de entrada de 32×32 píxeles, a diferencia de los 224×224 originales de ImageNet. Esto implica el uso de kernels más pequeños (3×3) y un stride de 1 en la primera capa convolucional para preservar la resolución espacial en las primeras etapas.

La red consta de cinco capas convolucionales y tres capas totalmente conectadas. Las primeras dos capas convolucionales son seguidas por capas de *Max Pooling*. Las capas convolucionales 3, 4 y 5 están conectadas directamente, seguidas de una última capa de *Max Pooling*. Las capas totalmente conectadas tienen 4096 neuronas cada una, y se utiliza *Dropout* con una probabilidad de 0.5 para regularización. La capa de salida se ajusta al número de clases del problema (10 tanto para CIFAR-10 como para MNIST).

2.3. GoogLeNet (Inception v1)

GoogLeNet, también conocida como Inception v1, fue propuesta por Szegedy et al. (2014) y ganó el desafío ILSVRC 2014. Su principal contribución fue el diseño de una arquitectura que aumenta significativamente la profundidad y el ancho de la red sin incrementar desproporcionadamente el costo computacional. Esto se logra mediante el módulo "Inception", que aplica convoluciones de diferentes tamaños (1×1, 3×3, 5×5) y operaciones de pooling en paralelo, concatenando sus salidas. Además, utiliza convoluciones de 1×1 para reducción de dimensionalidad antes de las operaciones costosas.

Otra característica importante es el uso de *Global Average Pooling* al final de la red en lugar de capas totalmente conectadas densas, lo que reduce drásticamente el número de parámetros y el riesgo de sobreajuste. Para facilitar el entrenamiento de una red tan profunda (22 capas), se introdujeron clasificadores auxiliares conectados a capas intermedias que inyectan gradiente adicional durante el entrenamiento.

2.3.1. Detalles de la Implementación

La implementación utilizada adapta la arquitectura original para entradas de 32×32. Las capas iniciales se han simplificado para evitar una reducción excesiva de la dimensión espacial antes de llegar a los módulos Inception. La red consta de 9 módulos Inception apilados secuencialmente.

Se incluyen dos clasificadores auxiliares conectados a las salidas de los módulos Inception 4a y 4d (en nuestra implementación, posicionados estratégicamente para combatir el desvanecimiento del gradiente). Durante el entrenamiento, la pérdida total es una suma ponderada de la pérdida principal y las pérdidas auxiliares. En inferencia, estos auxiliares se descartan. La clasificación final se realiza mediante un *Adaptive Average Pooling* seguido de una capa de *Dropout* (p=0.4) y una única capa lineal.

2.4. VGG

La arquitectura VGG, presentada por Simonyan and Zisserman (2014), se destaca por su simplicidad y uniformidad. A diferencia de AlexNet o GoogLeNet, VGG utiliza exclusivamente filtros convolucionales muy pequeños de 3×3 con un stride de 1 y padding de 1, junto con capas de *Max Pooling* de 2×2. La premisa fundamental es que apilar múltiples capas convolucionales con filtros pequeños permite emular el campo receptivo de filtros más grandes (e.g., dos capas de 3×3 equivalen a una de 5×5) pero con menos parámetros y más no-linealidades, lo que incrementa la capacidad discriminativa de la red.

Existen varias configuraciones de profundidad, siendo las más comunes VGG11, VGG13, VGG16 y VGG19, donde el número indica la cantidad de capas con pesos entrenables. Para este trabajo, hemos seleccionado la variante **VGG16**, que ha demostrado un excelente equilibrio entre rendimiento y costo computacional en diversas tareas de reconocimiento visual.

2.4.1. Detalles de la Implementación

Nuestra implementación de VGG16 sigue la configuración D del artículo original. La red se compone de 13 capas convolucionales y 3 capas totalmente conectadas. Las capas convolucionales están organizadas en 5 bloques, separados por operaciones de *Max Pooling*. El número de filtros se duplica después de cada operación de pooling, comenzando con 64 y llegando hasta 512 ($64 \rightarrow 128 \rightarrow 256 \rightarrow 512 \rightarrow 512$).

Para adaptar la red a nuestras imágenes de 32×32, se eliminó la necesidad de grandes reducciones espaciales iniciales. Después de pasar por todos los bloques convolucionales y de pooling, el mapa de características resultante tiene dimensiones de $512 \times 1 \times 1$. Esto se aplana y pasa a través de dos capas densas de 4096 neuronas cada una (con *Dropout* de 0.5) y finalmente a la capa de salida. Además, hemos incorporado *Batch Normalization* después de cada capa convolucional para acelerar la convergencia y estabilizar el entrenamiento, una mejora moderna respecto al diseño original.

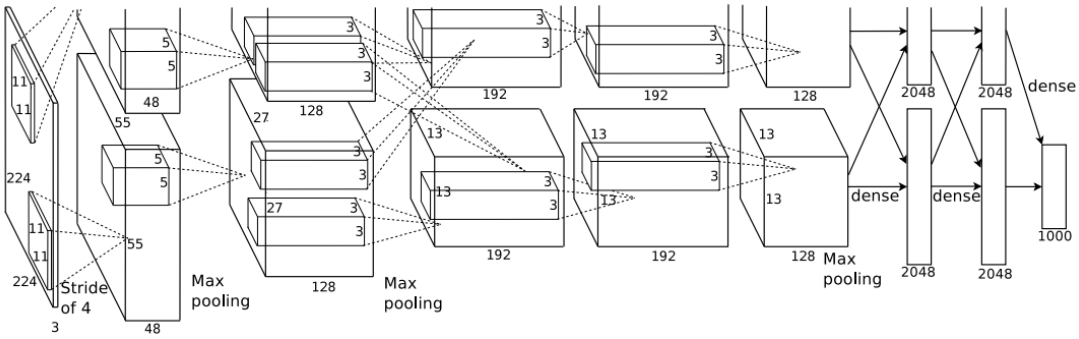

Figure 2. Arquitectura de la red AlexNet (Krizhevsky et al. 2012).

2.5. ResNet

ResNet (Residual Network), introducida por He et al. (2016), revolucionó el entrenamiento de redes neuronales extremadamente profundas al abordar el problema de la degradación del rendimiento. A medida que las redes aumentan en profundidad, el entrenamiento se vuelve más difícil debido al desvanecimiento del gradiente. ResNet soluciona esto mediante la introducción de "conexiones residuales" (skip connections), que permiten que la señal fluya directamente de una capa a otra saltándose capas intermedias.

La idea central es que es más fácil para la red aprender un mapeo residual $F(x) = H(x) - x$ que aprender el mapeo original $H(x)$ directamente. Si la identidad es la función óptima, la red puede simplemente llevar los pesos de las capas no lineales a cero. Esto permitió entrenar redes de cientos de capas (e.g., ResNet-152) con éxito.

Las arquitecturas ResNet se construyen apilando bloques residuales, de los cuales existen dos variantes principales:

- **BasicBlock:** Utilizado en modelos como ResNet-18 y ResNet-34. Consiste en dos capas convolucionales de 3×3 en serie.
- **Bottleneck Block:** Empleado en modelos más profundos (ResNet-50, 101, 152). Utiliza una estructura de tres capas: una convolución de 1×1 para reducir la dimensionalidad, una de 3×3 para el procesamiento, y finalmente una de 1×1 para restaurar la dimensión (expansión $\times 4$). Esto reduce el costo computacional permitiendo mayor profundidad.

2.5.1. Detalles de la Implementación

En este trabajo utilizamos **ResNet-18**, una variante más ligera adecuada para la complejidad de CIFAR-10 y MNIST. La arquitectura se basa en el "Bloque Básico" (BasicBlock), que consta de dos convoluciones, cada una seguida de *Batch Normalization* y ReLU. La conexión residual suma la entrada del bloque a la salida de la segunda convolución antes de la activación final.

Nuestra implementación comienza con una convolución inicial de 3×3 (adaptada de 7×7 para imágenes pequeñas) y luego apila cuatro "capas" residuales, cada una compuesta por dos bloques básicos. El número de filtros se duplica progresivamente: 64, 128, 256 y 512. Finalmente, se utiliza *Global Average Pooling* y una única capa totalmente conectada para la clasificación. Los pesos se inicializan utilizando el método de Kaiming He, optimizado para activaciones ReLU.

2.6. MobileNet

Las arquitecturas MobileNet fueron diseñadas específicamente para aplicaciones en dispositivos móviles y embebidos, donde los recursos computacionales (memoria y potencia de procesamiento) son limitados. El objetivo principal es lograr una alta precisión con un número reducido de parámetros y operaciones de punto flotante (FLOPs).

2.6.1. MobileNetV1

La primera versión de MobileNet Howard et al. (2017) introduce el concepto de **Convolución Separable en Profundidad** (Depthwise Separable Convolution). Esta operación descompone una convolución estándar en dos pasos más eficientes:

1. **Depthwise Convolution:** Aplica un filtro único por cada canal de entrada.
2. **Pointwise Convolution:** Aplica una convolución de 1×1 para combinar las salidas de la etapa anterior.

Esta factorización reduce drásticamente el costo computacional. Por ejemplo, con filtros de 3×3, el costo se reduce entre 8 y 9 veces en comparación con una convolución estándar, con una pérdida mínima de precisión.

2.6.2. MobileNetV2

MobileNetV2 Sandler et al. (2018) mejora la arquitectura original introduciendo el **Bloque Residual Invertido** (Inverted Residual Block). A diferencia de los bloques residuales tradicionales (que comprimen la dimensión en el medio), este bloque sigue una estructura "angosta $\rightarrow$ ancha $\rightarrow$ angosta":

1. Expande la dimensión de entrada con una convolución 1×1 (Expansion Layer).
2. Aplica una convolución depthwise de 3×3 en el espacio de alta dimensión.
3. Proyecta de nuevo a una dimensión baja con una convolución 1×1 (Projection Layer).

Además, elimina la función de activación no lineal (ReLU) después de la capa de proyección (Linear Bottleneck) para preservar la información, ya que las no-linealidades pueden destruir información en espacios de baja dimensión.

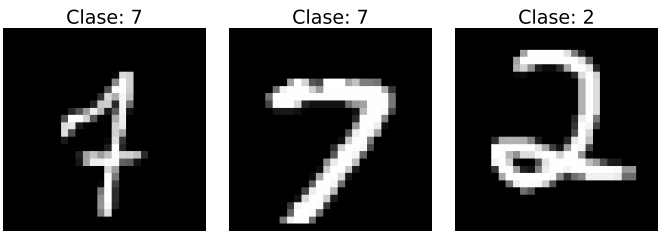

Figure 3. Ejemplos de imágenes del conjunto de datos MNIST.

Figure 4. Ejemplos de imágenes del conjunto de datos CIFAR-10.

2.6.3. Detalles de la Implementación

Para este trabajo, implementamos ambas versiones utilizando el multiplicador de ancho estándar de 1.0.

- **MobileNetV1:** Nuestra implementación consta de una capa convolucional inicial seguida de 13 bloques de convolución separable en profundidad. La red finaliza con un *Global Average Pooling* y una capa densa.
- **MobileNetV2:** La implementación utiliza una pila de bloques residuales invertidos con diferentes factores de expansión (t=1 o t=6). Se utiliza la función de activación ReLU6 $(\min(\max(0, x), 6))$ para mejorar la robustez en aritmética de baja precisión. La red termina con una capa de convolución 1×1 de 1280 canales antes del pooling global y la clasificación.

Ambas arquitecturas fueron adaptadas para aceptar imágenes de entrada de 32×32, ajustando los strides en las primeras capas para evitar una reducción prematura de la resolución espacial.

3. Conjuntos de datos

Para evaluar el desempeño de las arquitecturas implementadas, se utilizaron dos conjuntos de datos de referencia en la comunidad de aprendizaje automático.

3.1. MNIST

El conjunto de datos MNIST (Modified National Institute of Standards and Technology) (Deng 2012) es uno de los benchmarks más populares para algoritmos de clasificación de imágenes. Consiste en 70,000 imágenes en escala de grises de dígitos manuscritos del 0 al 9 (Figura 3). Las imágenes tienen una dimensión de 28×28 píxeles. El conjunto está dividido en 60,000 imágenes para entrenamiento y 10,000 para prueba. Aunque es considerado un problema "resuelto" para arquitecturas modernas, sigue siendo útil para verificar la corrección de las implementaciones y como línea base. En nuestros experimentos, las imágenes fueron redimensionadas (añadiendo padding) a 32×32 para mantener consistencia con la entrada de las arquitecturas diseñadas para CIFAR-10.

3.2. CIFAR-10

El conjunto de datos CIFAR-10 (Krizhevsky 2009) es una colección de imágenes etiquetadas que se utiliza comúnmente para entrenar algoritmos de visión por computadora. Contiene 60,000 imágenes a color de 32×32 píxeles divididas en 10 clases mutuamente excluyentes: avión, automóvil, pájaro, gato, ciervo, perro, rana, caballo, barco y camión (Figura 4). Hay 6,000 imágenes por clase. El conjunto se divide en 50,000 imágenes de entrenamiento y 10,000 de prueba. A diferencia de MNIST, CIFAR-10 presenta desafíos significativos debido a la variabilidad en la pose, iluminación, escala y fondo de los objetos, así como la baja resolución de las imágenes.

4. Experimentos y Resultados

References

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 2002.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. URL http://arxiv.org/abs/1409.4842.