Fundamentos Matemáticos de las Redes Neuronales Convolucionales

Pedro Elías Mirón Enríquez

29 de noviembre de 2024

Resumen

Este documento presenta un análisis riguroso de las redes neuronales convolucionales (CNN), comenzando desde los fundamentos matemáticos de la convolución hasta su aplicación en el procesamiento de imágenes digitales y el aprendizaje profundo.

1. Fundamentos de la Convolución

1.1. Definición Matemática

Definición 1.1 (Convolución Continua)**.** La convolución de dos funciones $f, g : \mathbb{R} \to \mathbb{R}$ se define como:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau$$

Definición 1.2 (Convolución Discreta)**.** Para secuencias discretas, la convolución se define como:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n - m]$$

1.2. Convolución en Imágenes 2D

Para imágenes digitales, utilizamos la convolución 2D discreta:

$$(I * K)[i, j] = \sum_{m} \sum_{n} I[m, n]K[i - m, j - n]$$

donde I es la imagen y K es el kernel de convolución.

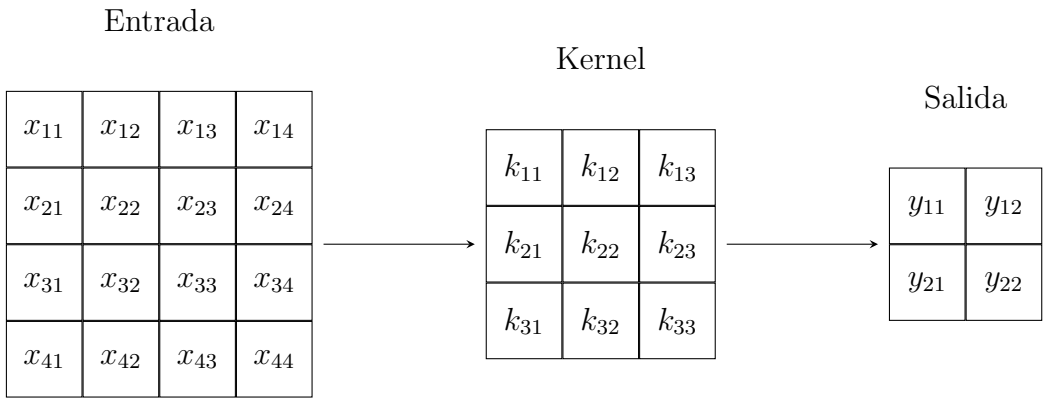

Figura 1: Operación de convolución 2D

2. Distinción Fundamental entre CNNs y Redes Neuronales Tradicionales

2.1. Conectividad Local y Compartición de Parámetros

Definición 2.1 (Conectividad Local)**.** Sea $\mathcal{N}(i,j)$ el vecindario de la posición (i,j) definido como:

$$\mathcal{N}(i,j) = \{(s,t) : |s-i| \leq k, |t-j| \leq k\}$$

donde k es el radio del kernel. La conectividad local implica que cada neurona en la capa l solo se conecta con las neuronas en $\mathcal{N}(i,j)$ de la capa $l-1$.

Teorema 2.1 (Reducción de Parámetros)**.** Para una capa convolucional con kernel de tamaño $K \times K$ y C_{in} canales de entrada y C_{out} canales de salida, el número de parámetros es:

$$P_{CNN} = K^2 \cdot C_{in} \cdot C_{out} + C_{out}$$

mientras que para una capa completamente conectada con las mismas dimensiones sería:

$$P_{FC} = (W \cdot H \cdot C_{in}) \cdot (W' \cdot H' \cdot C_{out}) + W' \cdot H' \cdot C_{out}$$

donde W, H son las dimensiones espaciales de entrada y W', H' las de salida.

Cuadro 1: Comparación entre FFNNs y CNNs

Aspecto	**FFNNs**	**CNNs**
Ventajas	▪ Implementación simple ▪ Versatilidad ▪ Menor costo computacional ▪ Efectivas con datos tabulares ▪ Mayor interpretabilidad	▪ Excelentes con datos espaciales ▪ Menos parámetros ▪ Invarianza a la traslación ▪ Pesos compartidos ▪ Detección jerárquica
Desventajas	▪ Mal manejo de datos espaciales ▪ Mayor número de parámetros ▪ Menor capacidad de abstracción ▪ No preservan relaciones espaciales ▪ Propensas a sobreajuste	▪ Implementación compleja ▪ Alto costo computacional ▪ Requieren más datos ▪ Específicas para datos estructurados ▪ Difíciles de interpretar

Demostración. En una CNN, cada filtro tiene K^2 parámetros y se aplica a todos los canales de entrada (C_{in}). Hay C_{out} filtros diferentes, más un término de bias por filtro.

En una capa FC, cada neurona de salida se conecta con cada neurona de entrada, resultando en el producto de las dimensionalidades totales. □

2.2. Invarianza Traslacional

Proposición 2.1 (Invarianza Traslacional)**.** Sea T_δ un operador de traslación por δ pixels. Una CNN con kernel K exhibe la propiedad:

$$K * (T_\delta x) \approx T_\delta (K * x)$$

Lema 2.1 (Equivarianza)**.** La operación de convolución es equivariante a la traslación:

$$K * (T_\delta x) = T_\delta (K * x)$$

Esta propiedad se mantiene exactamente para convoluciones circulares y aproximadamente para convoluciones con padding.

2.3. Comparación Matemática con Redes Fully-Connected

Una capa fully-connected realiza la siguiente operación:

$$y_j = \sigma \left(\sum_i W_{ji} x_i + b_j \right)$$

En contraste, una capa convolucional realiza:

$$y_{j,m,n} = \sigma \left(\sum_i \sum_{p,q} K_{j,i,p,q} x_{i,m+p,n+q} + b_j \right)$$

donde:

- $K_{j,i,p,q}$ es el kernel convolucional
- Los índices (p, q) recorren solo el tamaño del kernel
- Los pesos $K_{j,i,p,q}$ se comparten entre todas las posiciones espaciales

Figura 2: Comparación de conectividad entre redes FC y CNN

2.4. Propiedades Fundamentales que Distinguen las CNNs

1. **Sparse Interactions:** A diferencia de las capas fully-connected, cada neurona en una CNN interactúa solo con un subconjunto local de las entradas.

2. **Parameter Sharing:** Los mismos pesos del kernel se aplican a diferentes posiciones de la entrada, reduciendo significativamente el número de parámetros.

3. **Equivariant Representations:** Las CNNs son equivariantes a la traslación, lo que significa que un desplazamiento en la entrada resulta en un desplazamiento correspondiente en la salida.

Teorema 2.2 (Complejidad Computacional)**.** Para una capa convolucional con entrada de tamaño $n \times n$, kernel de tamaño $k \times k$, y m canales:

$$\text{Complejidad CNN} = O(k^2 m n^2)$$

mientras que para una capa fully-connected equivalente:

$$\text{Complejidad FC} = O(n^4)$$

3. Padding en Convoluciones

3.1. Valid Padding

Definición 3.1 (Valid Padding)**.** El valid padding es una estrategia de convolución donde solo se realizan operaciones cuando el kernel está completamente contenido dentro de la imagen, sin añadir valores adicionales en los bordes. Para una entrada de tamaño $(n \times n)$ y un kernel de tamaño $(k \times k)$, la salida tendrá dimensiones:

$$(n - k + 1) \times (n - k + 1)$$

Teorema 3.1 (Reducción de Dimensionalidad con Valid Padding)**.** Para una secuencia de L capas convolucionales con kernels de tamaño k_i, la dimensión de salida n_L dada una entrada n_0 es:

$$n_L = n_0 - \sum_{i=1}^{L}(k_i - 1)$$

Demostración. En cada capa i, la dimensión se reduce en $(k_i - 1)$. Aplicando esto secuencialmente:

$$\begin{aligned} n_1 &= n_0 - (k_1 - 1) \\ n_2 &= n_1 - (k_2 - 1) = n_0 - (k_1 - 1) - (k_2 - 1) \\ &\vdots \\ n_L &= n_0 - \sum_{i=1}^{L}(k_i - 1) \end{aligned}$$

□

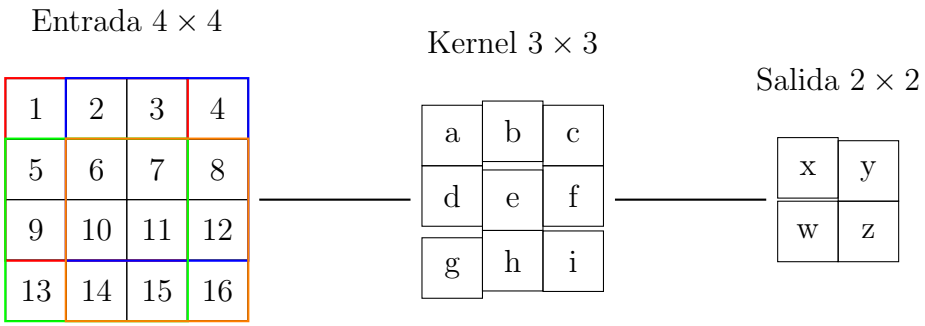

Figura 3: Ejemplo de Valid Padding: La convolución de una matriz 4×4 con un kernel 3×3 produce una salida 2×2. Las regiones coloreadas muestran las cuatro posibles posiciones válidas del kernel.

Lema 3.1 (Preservación de Información)**.** El valid padding garantiza que cada valor en la salida se compute utilizando únicamente información real de la entrada, sin valores artificiales añadidos.

3.2. Comparación con Otros Tipos de Padding

Tipo de Padding	Dimensión de Salida	Característica Principal
Valid	$n - k + 1$	No añade valores adicionales
Same	n	Mantiene dimensiones originales
Full	$n + k - 1$	Considera todas las superposiciones posibles

Cuadro 2: Comparación de diferentes estrategias de padding

Proposición 3.1 (Ventajas y Desventajas del Valid Padding)**.** El valid padding ofrece:

- **Ventajas:**
  - No introduce artefactos artificiales
  - Garantiza cálculos basados solo en datos reales
  - Reduce la dimensionalidad de forma natural
- **Desventajas:**
  - Pérdida de información en los bordes
  - Reducción significativa del tamaño de salida
  - Posible pérdida de características importantes en los bordes

Corolario 3.1.1 (Conservación de Bordes)**.** Para conservar información de los bordes con valid padding, la entrada debe tener un tamaño mínimo de:

$$n_{min} = k + (d - 1)$$

donde k es el tamaño del kernel y d es la dimensión mínima deseada de salida.

4. Representación Matricial de la Convolución

4.1. Matriz de Toeplitz y Tipos de Padding

Definición 4.1 (Matriz de Toeplitz para Convolución 1D)**.** Dado un kernel $k = [k_1, k_2, k_3]$, la matriz de Toeplitz correspondiente para diferentes tipos de padding es:

Valid Padding:

$$T_{valid} = \begin{bmatrix} k_1 & k_2 & k_3 & 0 & 0 \\ 0 & k_1 & k_2 & k_3 & 0 \\ 0 & 0 & k_1 & k_2 & k_3 \end{bmatrix}$$

Same Padding:

$$T_{same} = \begin{bmatrix} k_2 & k_3 & 0 & 0 & k_1 \\ k_1 & k_2 & k_3 & 0 & 0 \\ 0 & k_1 & k_2 & k_3 & 0 \\ 0 & 0 & k_1 & k_2 & k_3 \\ k_3 & 0 & 0 & k_1 & k_2 \end{bmatrix}$$

Full Padding:

$$T_{full} = \begin{bmatrix} k_3 & 0 & 0 & k_1 & k_2 \\ k_2 & k_3 & 0 & 0 & k_1 \\ k_1 & k_2 & k_3 & 0 & 0 \\ 0 & k_1 & k_2 & k_3 & 0 \\ 0 & 0 & k_1 & k_2 & k_3 \\ k_3 & 0 & 0 & k_1 & k_2 \\ k_2 & k_3 & 0 & 0 & k_1 \end{bmatrix}$$

4.2. Extensión a 2D: Matrices de Toeplitz-Block

Teorema 4.1 (Convolución 2D como Multiplicación de Matrices)**.** Una convolución 2D puede expresarse como una multiplicación de matrices:

$$y = Tx$$

donde T es una matriz de Toeplitz-block y x es la imagen vectorizada.

Para un kernel $K \in \mathbb{R}^{k \times k}$ y una imagen $I \in \mathbb{R}^{n \times n}$:

$$T = \begin{bmatrix} T_{11} & T_{12} & \cdots & T_{1n} \\ T_{21} & T_{22} & \cdots & T_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ T_{n1} & T_{n2} & \cdots & T_{nn} \end{bmatrix}$$

donde cada T_{ij} es una matriz de Toeplitz que depende del tipo de padding.

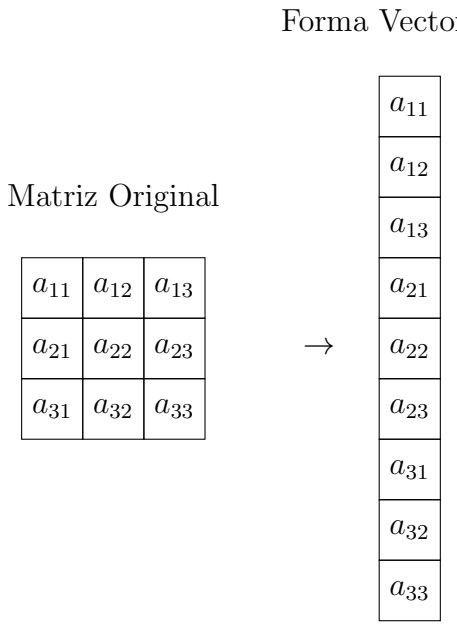

Figura 4: Vectorización de la matriz de entrada

5. Impacto del Padding en las Operaciones Matriciales

5.1. Valid Padding

Para valid padding, la matriz de multiplicación tiene dimensiones:

$$T_{valid} \in \mathbb{R}^{((n-k+1)(n-k+1))\times(n^2)}$$

$$\begin{bmatrix} K_{11} & K_{12} & K_{13} & 0 & \cdots & 0 \\ 0 & K_{11} & K_{12} & K_{13} & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & K_{11} & K_{12} & K_{13} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n^2} \end{bmatrix}$$

5.2. Same Padding

La matriz para same padding tiene dimensiones:

$$T_{same} \in \mathbb{R}^{(n^2)\times(n^2)}$$

El padding se refleja en las filas adicionales de la matriz T:

$$\begin{bmatrix} K_{11} & 0 & \cdots & K_{13} & K_{12} \\ K_{12} & K_{11} & 0 & \cdots & K_{13} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ K_{13} & \cdots & 0 & K_{11} & K_{12} \\ K_{12} & K_{13} & \cdots & 0 & K_{11} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n^2} \end{bmatrix}$$

Proposición 5.1 (Eficiencia Computacional)**.** La representación matricial explícita requiere:

- Valid Padding: $O((n-k+1)^2 n^2)$ elementos
- Same Padding: $O(n^4)$ elementos
- Full Padding: $O((n+k-1)^2 n^2)$ elementos

6. Im2col: Transformación Eficiente

Definición 6.1 (Operación Im2col)**.** La operación im2col transforma una imagen y un kernel en matrices que pueden multiplicarse eficientemente:

$$Y = \text{reshape}((K_{flat} \cdot \text{im2col}(X))^T)$$

donde:

- $K_{flat} \in \mathbb{R}^{c_{out} \times (k^2 c_{in})}$
- $\text{im2col}(X) \in \mathbb{R}^{(k^2 c_{in}) \times hw}$
- h, w son las dimensiones espaciales de salida

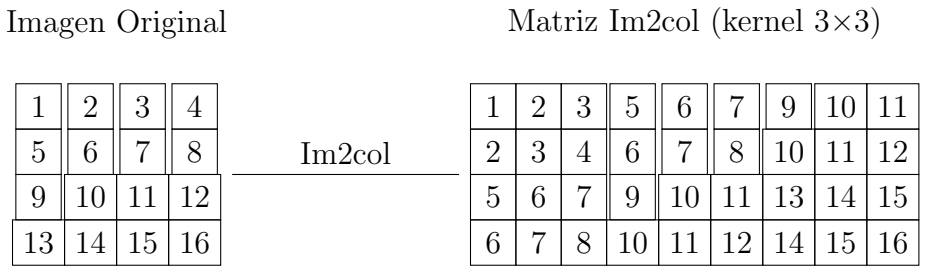

Figura 5: Transformación Im2col para convolución eficiente

Corolario 6.0.1 (Complejidad de Im2col)**.** La transformación im2col reduce la complejidad de la convolución a una única multiplicación de matrices, pero aumenta el uso de memoria en un factor de k^2.

7. Arquitectura de CNNs

7.1. Capas Convolucionales

La salida de una capa convolucional para el canal k se define como:

$$h_k^l = \sigma\left(\sum_{i=1}^{C_{l-1}} W_{k,i}^l * x_i^{l-1} + b_k^l\right)$$

donde:

- h_k^l es la k-ésima feature map en la capa l
- $W_{k,i}^l$ es el kernel convolucional
- x_i^{l-1} es la i-ésima feature map de entrada
- b_k^l es el término de bias
- σ es la función de activación

7.2. Pooling

Definición 7.1 (Max Pooling)**.** El max pooling sobre una región $\mathcal{R}$ se define como:

$$p(x)_{i,j} = \max_{(s,t)\in\mathcal{R}_{i,j}} x_{s,t}$$

8. Retropropagación en CNNs

Teorema 8.1 (Gradiente de la Convolución)**.** El gradiente de la función de pérdida L con respecto al kernel W está dado por:

$$\frac{\partial L}{\partial W} = \sum_{i,j} \frac{\partial L}{\partial y_{i,j}} \frac{\partial y_{i,j}}{\partial W}$$

donde $y_{i,j}$ es la salida de la convolución.

Demostración. Por la regla de la cadena y la definición de convolución:

$$\frac{\partial y_{i,j}}{\partial W_{a,b}} = x_{i+a,j+b}$$

Por lo tanto:

$$\frac{\partial L}{\partial W_{a,b}} = \sum_{i,j} \frac{\partial L}{\partial y_{i,j}} x_{i+a,j+b}$$

□

9. Ejemplos Clásicos

9.1. LeNet-5

Arquitectura pionera desarrollada por Yann LeCun para el reconocimiento de dígitos manuscritos.

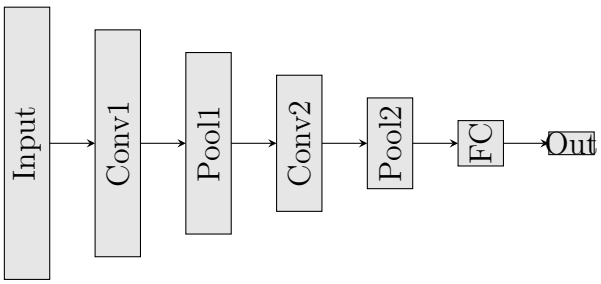

Figura 6: Arquitectura LeNet-5

9.2. AlexNet

Arquitectura que popularizó las CNNs al ganar ImageNet en 2012.

10. Referencias

1. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

2. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. NIPS.

3. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.