Deep Learning

Episodio 5: Redes Neuronales Convolucionales

Fernando Gama

Escuela de Graduados en Ingeniería Informática y Sistemas, Facultad de Ingeniería, UBA

4 de Agosto de 2022

Escenas del capítulo anterior...

- A veces, los datos tienen estructuras que se pueden explotar para mejorar el algoritmo
  $\Rightarrow$ Señales en el tiempo, sistemas para procesarlos
- Operación de convolución $\Rightarrow$ Operación lineal que explota la estructura
- Respuesta en frecuencia $\Rightarrow$ Representación alternativa de señales con estructura
  $\Rightarrow$ Permite encontrar una manera rápida y fácil de calcular la convolución
- Convolución en imágenes $\Rightarrow$ Conceptualmente igual que la convolución en el tiempo

Redes Neuronales Convolucionales

Pooling

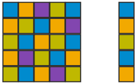
Arquitecturas

Redes Neuronales Completas

- Las redes neuronales son una cascada de capas $\Rightarrow$ Función lineal seguido de activación

$$\mathbf{x}_\ell = \sigma_\ell\left(\mathbf{W}_\ell \mathbf{x}_{\ell-1} + \mathbf{b}_\ell\right)$$

- Si la dimensión N de los datos es muy grande $\Rightarrow$ $\mathbf{W}_0$ va a ser muy grande
  $\Rightarrow$ Más parámetros $\Rightarrow$ Es más difícil de aprender $\Rightarrow$ Necesito más datos



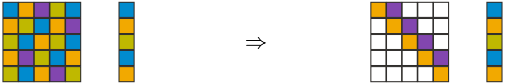
- Si los datos no tienen estructura, no queda otra que conseguir más datos
  $\Rightarrow$ Pero si tienen estructura, podemos elegir la operación lineal de manera más inteligente

Estructura de los Datos

- Si los datos tienen una estructura regular (datos contiguos están relacionados)
  - $\Rightarrow$ Podemos usar la operación de convolución (es una operación lineal) en lugar de un $\mathbf{W}_\ell$ genérico

- Observar que, ahora, la transformación lineal sólo relaciona elementos contiguos
  - $\Rightarrow$ Es lo mismo que hace la convolución: combinación lineal de pocos elementos y desplazamiento

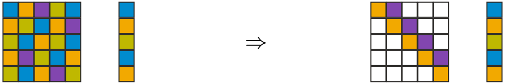
- Informalmente, una red neuronal convolucional (CNN) es una perceptrón multicapa (MLP)
  - $\Rightarrow$ Se reemplaza la transformación lineal por una convolución (que también es lineal)
  - $\Rightarrow$ La operación de convolución se escribe como una matriz con estructura Toeplitz



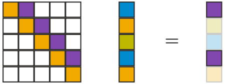
- Ahora no hay que aprender $N \times N_1$ parámetros en $\mathbf{W}_0$, sino que hay que aprender $K \ll N$
- Las redes neuronales convolucional han sido tremendamente exitosas
  - $\Rightarrow$ Fundamentalmente para procesar imágenes (clasificación, detección, etc.)

Redes Neuronales Convolucionales

- Las CNNs son un subconjunto de las MLPs $\Rightarrow$ Sin embargo funcionan mejor (en imágenes)



- Son más fáciles de optimizar $\Rightarrow$ Menos parámetros para aprender, menos datos
- Son más computacionalmente eficientes $\Rightarrow$ Requieren $\mathsf{O}(KN)$ operaciones en vez de $\mathsf{O}(N^2)$
- Comparten parámetros $\Rightarrow$ Reducen el almacenamiento en memoria: $\mathsf{O}(K)$ vs. $\mathsf{O}(N^2)$
- Usan la estructura de los datos para generalizar mejor $\Rightarrow$ Equivarianza a traslaciones



- Si los datos se repiten $\Rightarrow$ La salida se repite
- La CNN captura información independientemente de su ubicación

- Una transformación lineal arbitraria puede cambiar las dimensiones
  - $\Rightarrow$ La matriz $\mathbf{W}_\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ toma un vector en $N_{\ell-1}$ y devuelve un vector en N_ℓ
- Las convoluciones toman una señal de entrada, y devuelven una señal de salida
  - $\Rightarrow$ La convolución toma un vector de dimensión N y devuelve un vector en $N - K + 1$
- Al utilizar una convolución, perdimos la capacidad de ajustar las dimensiones
  - $\Rightarrow$ Perdimos la posibilidad de ajustar la capacidad de representación del algoritmo

Características (*Features*) o Canales (*Channels*)

- Se puede recuperar el control de la capacidad de representación usando muchas convoluciones
  - ⇒ Entra la señal $\mathbf{x}$ y la filtramos con F_1 convoluciones h_1^f con K_1 coeficientes

$$\mathbf{x} \mapsto \{\mathsf{h}_1^1 * \mathbf{x} + \mathsf{b}_1^1, \mathsf{h}_1^2 * \mathbf{x} + \mathsf{b}_1^2, \ldots, \mathsf{h}_1^{F_1} * \mathbf{x} + \mathsf{b}_1^{F_1}\} = \{\mathbf{z}_1^1, \ldots, \mathbf{z}_1^{F_1}\}$$

- Las podemos juntar en una matriz $\mathbf{Z}$ de tamaño $N - K_1 + 1 \times F_1$

$$\mathbf{Z} = \begin{bmatrix} \mathbf{z}^1 & \mathbf{z}^2 & \cdots & \mathbf{z}^{F_1} \end{bmatrix}$$

- Ahora aplicamos la función de activación $\mathbf{X}_1 = \sigma_1(\mathbf{Z})$
  - ⇒ La salida de la primer capa tiene tamaño $N - K_1 + 1 \times F_1$
  - ⇒ El valor de F_1 lo controlamos (se elige por diseño) ⇒ Controla la capacidad de representación

- Cada vector $\mathbf{x}_1^f$ (cada columna de $\mathbf{X}_1$) es un *feature* (o *channel*)
  - ⇒ F_1 es la cantidad de features (o la cantidad de channels) de la capa $\ell = 1$
  - ⇒ La cantidad de parámetros a aprender ahora es $K_1 F_1$ (indep. de N)

Características (*Features*) o Canales (*Channels*)

- A la salida de la primer capa, tenemos F_1 features $\mathbf{x}_1^f$ en la matriz $\mathbf{X}_1 \in \mathbb{R}^{N-K_1+1 \times F_1}$
  - $\Rightarrow$ Si ahora quiero F_2 features a la salida de la segunda capa, ¿cómo hacemos?
  - $\Rightarrow$ Recordar que en un MLP la dimensión N_ℓ (features) de cada capa era elección del diseñador
- En la capa 2 se usan $F_1 \times F_2$ convoluciones con filtros h_2^{fg} con K_2 coeficientes $\Rightarrow$ Banco de filtros

$$\{\mathbf{x}_1^1, \ldots, \mathbf{x}_1^{F_1}\} \mapsto \begin{Bmatrix} \mathsf{h}_2^{11} * \mathbf{x}_1^1, & \mathsf{h}_2^{12} * \mathbf{x}_1^1, & \ldots, & \mathsf{h}_2^{1F_2} * \mathbf{x}_1^1, \\ \mathsf{h}_2^{21} * \mathbf{x}_1^2, & \mathsf{h}_2^{22} * \mathbf{x}_1^2, & \ldots, & \mathsf{h}_2^{2F_2} * \mathbf{x}_1^2, \\ \ldots, & & & \\ \mathsf{h}_2^{F_1 1} * \mathbf{x}_1^{F_1}, & \mathsf{h}_2^{F_1 2} * \mathbf{x}_1^{F_1}, & \ldots, & \mathsf{h}_2^{F_1 F_2} * \mathbf{x}_1^{F_1} \end{Bmatrix}$$

- Esto da $F_1 F_2$ señales a la salida $\Rightarrow$ Cada señal de tamaño $N - K_1 - K_2 + 2$

$$\{\mathbf{x}_1^1, \ldots, \mathbf{x}_1^{F_1}\} \mapsto \begin{Bmatrix} \mathbf{z}_2^{11}, & \mathbf{z}_2^{12}, & \ldots, & \mathbf{z}_2^{1F_2}, \\ \mathbf{z}_2^{21}, & \mathbf{z}_2^{22}, & \ldots, & \mathbf{z}_2^{2F_2}, \\ \ldots, & & & \\ \mathbf{z}_2^{F_1 1}, & \mathbf{z}_2^{F_1 2}, & \ldots, & \mathbf{z}_2^{F_1 F_2} \end{Bmatrix}$$

  - $\Rightarrow$ Pero queremos sólo F_2 señales, no $F_1 F_2$ señales $\Rightarrow$ ¿Qué hacemos?

- Tenemos $F_1 F_2$ señales $\mathbf{z}_2^{fg}$, pero queremos sólo F_2 señales

  $\Rightarrow$ Sumamos las señales en la dimensión F_1 $\Rightarrow \mathbf{z}_2^{g} = \sum_{f=1}^{F_1} \mathbf{z}_2^{fg}$

$$
\begin{array}{c|c|c|c}
\mathbf{z}_2^{11} & \mathbf{z}_2^{12} & \cdots & \mathbf{z}_2^{1F_2} \\
+\ \mathbf{z}_2^{21} & +\ \mathbf{z}_2^{22} & \cdots & +\ \mathbf{z}_2^{2F_2} \\
\vdots & \vdots & \ddots & \cdots \\
+\ \mathbf{z}_2^{F_1 1} & +\ \mathbf{z}_2^{F_1 2} & \cdots & +\ \mathbf{z}_2^{F_1 F_2} \\
\hline
= & = & = & = \\
\mathbf{z}_2^{1} & \mathbf{z}_2^{2} & \cdots & \mathbf{z}_2^{F_2}
\end{array}
$$

- Operación lineal, que explota la estructura y que ofrece control sobre la representación

  $\Rightarrow$ Convolución = Banco de filtros + Suma $\Rightarrow$ Mapea F_1 features en F_2

Características (*Features*) o Canales (*Channels*)

- Recuperamos el control sobre la capacidad de representación de la arquitectura
  - $\Rightarrow$ Convolución = Banco de filtros + Suma sobre la dimensión de entrada
  - $\Rightarrow$ El diseñador elige la cantidad de features en cada capa F_ℓ

Red Neuronal Convolucional

$$\mathsf{f}(\mathbf{X};\boldsymbol{\Theta}) = \{\mathbf{x}_L^g\}_{g=1}^{F_L} \quad \text{con} \quad \mathbf{x}_\ell^g = \sigma_\ell\left(\sum_{f=1}^{F_{\ell-1}} \mathsf{h}_\ell^{fg} * \mathbf{x}_{\ell-1}^f + \mathbf{b}_\ell^g\right),\ \ell = 1,\ldots,L \quad \text{y} \quad \mathbf{X}_0 = \mathbf{X}$$

- La señal de salida $\mathbf{X}_\ell$ en cada capa tiene dimensión $N - \sum_{\ell'=1}^{\ell} K_{\ell'} + \ell \times F_\ell$
- Los parámetros a aprender son los coef. de todos los filtros $\sum_{\ell=1}^{L} K_\ell F_\ell F_{\ell-1}$ $\Rightarrow$ Independiente de N
- Los hiperparámetros son el tamaño del filtro en cada capa K_ℓ y la cant. de features F_ℓ (y L y σ_ℓ)
- La convolución es lineal $\Rightarrow$ Puedo usar backpropagation como siempre

Redes Neuronales Convolucionales

Pooling

Arquitecturas

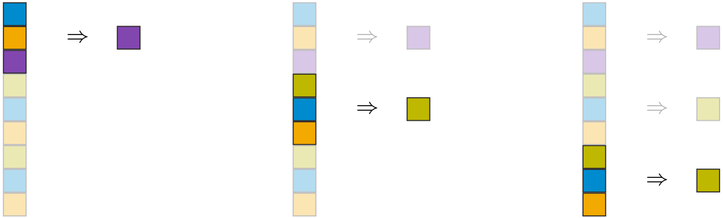
Pooling

- La señal de salida $\mathbf{X}_\ell$ en cada capa tiene dimensión $N - \sum_{\ell'=1}^{\ell} K_{\ell'} + \ell \times F_\ell$
- Con F_ℓ controlamos la capacidad de representación $\Rightarrow$ Mayor F_ℓ, mayor capacidad
- Pero a medida que F_ℓ se hace más grande, el tamaño de las señales $\mathbf{X}$ se hace más grande

  $\Rightarrow$ El costo computacional se incrementa, la necesidad de almacenamiento en memoria también

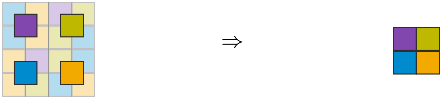
- La técnica de *pooling* se utiliza para construir resúmenes y reducir la dimensión de la señal

- Pooling $\Rightarrow$ Construir un resumen de una región contigua de valores de la señal
  $\Rightarrow$ Tomar un representante de la región, descartar los demás, y moverse a la región siguiente



- En el caso de imágenes, la idea es la misma: construir un resumen, tomar un representante

- El resultado de resumir las regiones es otra imagen, pero más chica



Resúmenes de las Regiones

- Para construir los resúmenes, aplicamos una determinada función a la región

- Las elecciones típicas son el máximo o el promedio de los valores en la región
  $\Rightarrow$ Pero puede haber muchas alternativas, como la norma o algún otro momento

- La elección de la función de resumen impacta en el desempeño del algoritmo
  $\Rightarrow$ Tanto la elección de la función como el tamaño de la región corren por cuenta del diseñador

Invarianza Local

- El uso de pooling le otorga a la arquitectura la propiedad de invarianza local (en la región)
  - ⇒ Sin importar dónde esté la información dentro de la región, igual la vamos a procesar
  - ⇒ Importante en problemas de clasificación donde la ubicación del concepto no es importante

- El uso de pooling transforma la equivarianza a traslaciones de la convolución
  - ⇒ En invarianza local a traslaciones
- Es otra forma de explotar la estructura de los datos para mejorar la generalización

Observaciones sobre el Pooling

- Pooling $\Rightarrow$ Nuevas decisiones para el diseñador: el tamaño de la región y la función de resumen

- El pooling aplica sólo a estructuras de datos donde la noción de región está bien definida
  $\Rightarrow$ Con la convolución son formas explotar la estructura de los datos para mejorar la generalización

- Usar pooling sirve para intercambiar información espacial (en la dimensión N)
  $\Rightarrow$ Por información aprendida en forma de features (en la dimensión F_ℓ)
  $\Rightarrow$ La idea es que $N_L F_L < N F_0$ $\Rightarrow$ Más features en imágenes más chicas

- Hay alternativas donde se hace pooling sobre la dimensión F $\Rightarrow$ Pero no tienen estructura regular

- Hacer pooling permite trabajar con señales de distinto tamaño

Redes Neuronales Convolucionales

Pooling

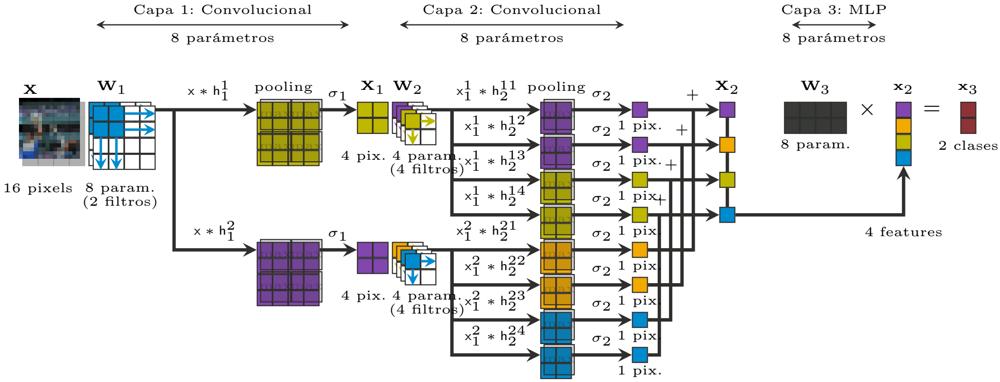
Arquitecturas

- Una red neuronal convolucional (CNN) es una arquitectura por capas
  - $\Rightarrow$ Donde cada capa es una convolución seguida de una función de activación
  - $\Rightarrow$ La convolución se realiza con un banco de filtros para controlar la capacidad de representación
  - $\Rightarrow$ Se puede incluir pooling para controlar dimensionalidad de las señales
- Se usan con datos de estructura regular $\Rightarrow$ Elementos contiguos están relacionados

- Los hiperparámetros a elegir por el diseñador son
  - $\Rightarrow$ La cantidad de capas L y las funciones de activación σ_ℓ
  - $\Rightarrow$ La cantidad de features a la salida de cada capa F_ℓ
  - $\Rightarrow$ El tamaño de los filtros K_ℓ
  - $\Rightarrow$ Si se incluye pooling: el tamaño de la región y la función de resumen
- La cantidad de parámetros a aprender es $\sum_\ell K_\ell F_\ell F_{\ell-1}$ $\Rightarrow$ Independiente de N

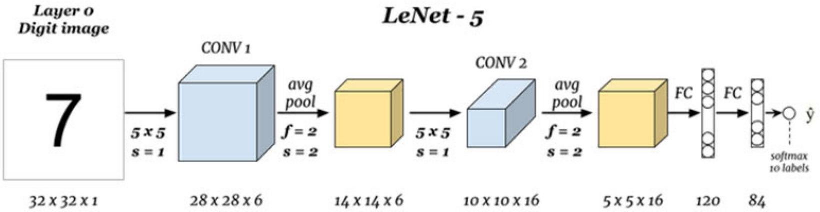
- Las CNNs han sido sumamente exitosas en procesamiento de imágenes
  - $\Rightarrow$ Problemas de clasificación: la entrada es una imagen, la salida una clase
  - $\Rightarrow$ Problemas de detección: la entrada es una imagen, la salida es una imagen
- En problemas donde la entrada es una imagen, pero la salida no (clasificación)
  - $\Rightarrow$ Se suele incluir MLPs luego de varias capas convolucionales
- Las capas convolucionales actúan como 'extractores de features'
- Las capas de MLPs actúan como 'clasificadores'
- El intercambio de dimensiones N_ℓ con features F_ℓ es importante
- *Stride* $\Rightarrow$ Salto en el cálculo de la convolución en una posición $\Rightarrow \mathsf{h} * \mathsf{x} = \sum_{k=0}^{K-1} h_k \, \mathsf{x}(n - s\, k)$

Arquitecturas



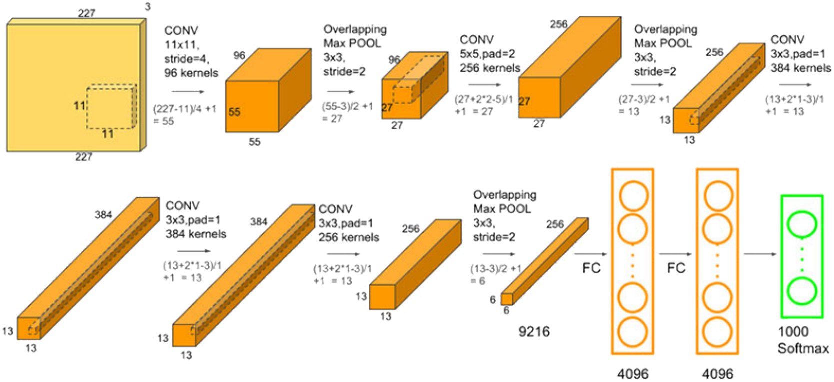
Arquitecturas: LeNet (LeCun, 1998)

- Una de las primeras en lograr un desempeño extraordinario en MNIST
- Usa pooling en promedio en vez de max pooling (en aquella época no se usaba max pooling)
- Usa tanh como la función de activación (todavía las ReLU no se habían popularizado)
- Hay diez dígitos posibles $\Rightarrow$ La última capa es un vector de tamaño 10



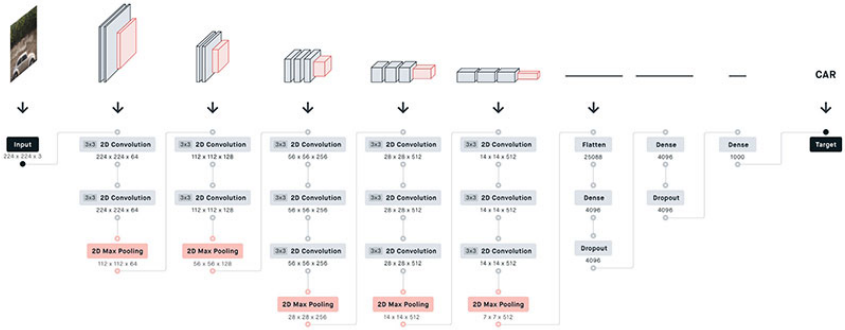
Arquitecturas: AlexNet (Krizhevsky, Sutskever, Hinton, 2012)

- Competición ImageNet $\Rightarrow$ Clasificación de imágenes en 1000 categorías $\Rightarrow$ 83 % en top-5



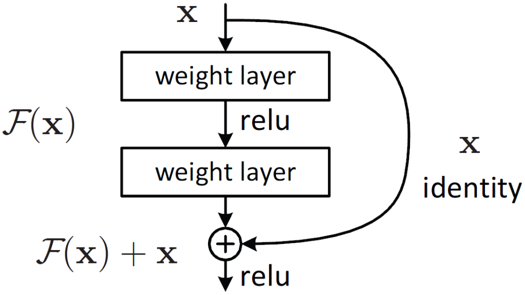
Arquitecturas: VGG (Simonyan, Zisserman, 2014)

- Competición ImageNet $\Rightarrow$ Clasificación de imágenes en 1000 categorías $\Rightarrow$ 92,7 % en top-5



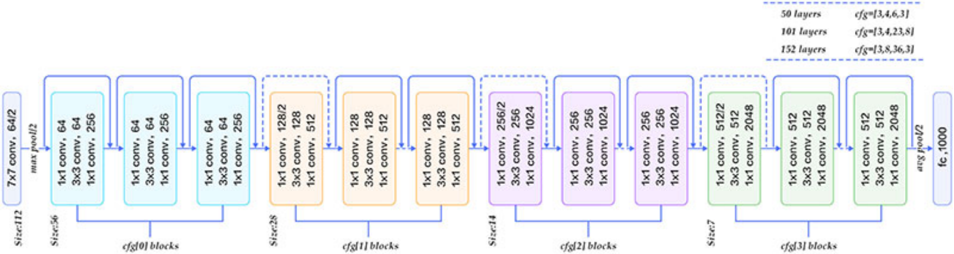
Arquitecturas: *Skip Connections*

- Luego de una determinada cantidad de capas, volver a incluir la entrada
  - ⇒ Evita que la información se diluya a través de las capas y los gradientes
  - ⇒ Particularmente importante para arquitecturas profundas



Arquitecturas: ResNets (He et al, 2015)

- Muy profunda ($L = 152$ capas) $\Rightarrow$ Resultado en ImageNet 96,4 % en top-5



Resumen del Capítulo

- Redes Neuronales Convolucionales $\Rightarrow$ Reemplazar la transformación lineal por una convolución
  - $\Rightarrow$ Se pierde el control sobre la capacidad de representación
  - $\Rightarrow$ Banco de Filtros + Suma

- Pooling $\Rightarrow$ Construir resúmenes regionales y luego quedarse sólo con un representante
  - $\Rightarrow$ Controlar el tamaño de las señales $\Rightarrow$ Invarianza local a traslaciones

- Arquitecturas típicas $\Rightarrow$ Incluir MLPs al final, conexiones residuales, etc.

Escenas del Próximo Capítulo...

- Queremos procesar secuencias de datos
  - $\Rightarrow$ Adaptamos la red neuronal para generalizar en datos secuenciales
  - $\Rightarrow$ Compartir parámetros $\Rightarrow$ Mismos valores durante toda la secuencia
- Redes neuronales recurrentes $\Rightarrow$ Aprenden un estado oculto
  - $\Rightarrow$ Este estado es capaz de capturar la información temporal relevante
- Área activa de investigación $\Rightarrow$ Muchas extensiones
  - $\Rightarrow$ Redes neuronales bidireccionales $\Rightarrow$ El contexto importa
  - $\Rightarrow$ LSTMs, GRUs $\Rightarrow$ Poder aprender dependencias de largo alcance