



Tecnológico  
de Monterrey

## Inteligencia Artificial Avanzada para la Ciencia de Datos

Daniel Otero Fadul

*Departamento de Matemáticas y Ciencia de Datos  
Escuela de Ingeniería y Ciencias*

El área de “**deep learning**” existe gracias a importantes avances que se empezaron a lograr desde los cincuentas, siendo el primero de estos el **perceptrón**. Sin embargo, Minsky, un profesor del MIT mostró que emular una compuerta XOR no era posible y declaró que redes neuronales no era un camino prometedor, dando comienzo al primer “AI Winter”.

Como vimos, sí es posible entrenar una red neuronal para que se comporte como una compuerta XOR gracias a **backpropagation**, algoritmo descubierto de manera independiente por varios investigadores en los sesenta, pero que tuvo que esperar hasta los setenta para ser utilizado en el entrenamiento de las redes neuronales, y hasta 1986 para ser popularizado por Rumelhart, Hinton y Williams.

En 1989 se demuestra que las redes neuronales son “aproximadores universales”, es decir, pueden aproximar cualquier tipo de función bajo ciertas condiciones. Es más, este mismo año Yann LeCun logra reconocer dígitos escritos a mano al implementar lo que conocemos actualmente como una red neuronal convolucional (CNN).

En los años siguientes se exploraron distintos tipos de arquitecturas para distintas aplicaciones: “autoencoders”, “belief nets”, “recurrent neural nets”, entre otras. Sin embargo, el entrenamiento de redes de una decena de capas con “backpropagation” ya resultaba poco práctico. Esto, sumado al hecho de que modelos nuevos y más sencillos de implementar como “Support Vector Machine” y “Random Forest”, que además obtenían resultados que eran el estado del arte en ese momento, propiciaron la llegada de un segundo “AI Winter” en la segunda mitad de los noventa.

El interés en las redes neuronales renace gracias a un artículo publicado en el 2006 por Hinton, Osindor y Yee-Whye titulado “A fast learning algorithm for deep belief nets”. En este trabajo mostraron que redes neuronales con muchas capas sí podían ser entrenadas efectivamente. Sin embargo, debido a la mala fama que tenían las redes neuronales en aquel entonces, decidieron llamar a este “nuevo” enfoque **“deep learning”** para que su trabajo fuera publicado. Su propuesta logró resultados que fueron considerados el estado del arte en aquel entonces con la base de datos **MNIST** (<https://deeppai.org/dataset/mnist>).

Sin embargo, tener buenos resultados con la base de datos MNIST no era suficiente pues resultados similares se podían obtener con SVM. Así que Hinton y dos de sus estudiantes de doctorado decidieron utilizar sus “deep belief nets” en otra aplicación más desafiante: reconocimiento de voz. El que obtuvieran mejores resultados de los anteriores obtenidos en esta área convenció a la comunidad de IA que las redes neuronales merecían más atención de la que les habían dado.

Era claro que redes neuronales de muchas capas podían aprender representaciones bastante complejas de los datos, pero para lograr esto se necesita conjuntos de entrenamiento de considerable tamaño. Debido a esto, Fei-Fei Li decide enfocarse en la recolección de una gran cantidad de imágenes para entrenar modelos de IA, en vez de enfocarse en los modelos, lo cual era la tendencia del momento. Su proyecto se convirtió en la base de datos conocida como **ImageNet**.

Gracias a este proyecto, se decide crear un reto de reconocimiento de imágenes utilizando los datos almacenados en ImageNet. En el 2010 se organiza por primera vez el **“ImageNet Large Scale Visual Recognition Challenge”** (ILSVRC).

La pieza que faltaba en el rompecabezas de “deep learning” era entrenar redes neuronales con el poder computacional de las **unidades de procesamiento gráfico**, también conocidas como GPUs por sus siglas en inglés. Mohamedy Dahl, dos estudiantes de doctorado de Hinton, lograron obtener muy buenos resultados entrenando redes neuronales de muchas capas con la ayuda de GPUs. Esto demostró que el poder computacional juega un papel importante en el rendimiento de las redes neuronales.

# Deep Learning

En el 2012 Hinton se inscribe en el reto de ImageNet, ILSVRC, junto con sus coautores del método “dropout”, Alex Krizhevsky y Ilya Sutskever. Era la primera vez que una red convolucional participaba, es más, era la única red neuronal participando en el reto. Ganaron el primer lugar de manera bastante holgada: su taza de error reconociendo imágenes fue del 15.3%, mientras que la taza de error del segundo lugar fue 26.2%. La red neuronal ganadora es la famosa **AlexNet**.



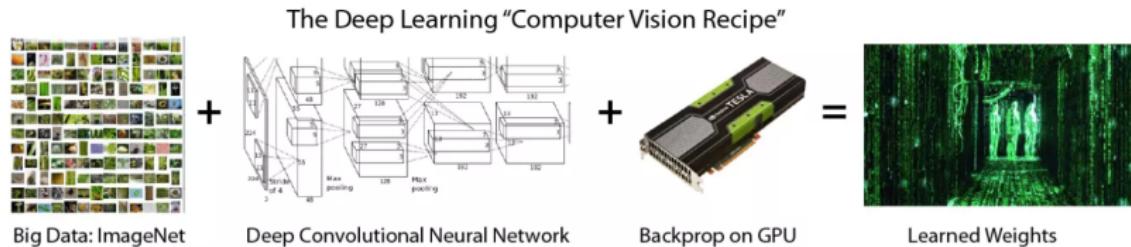
**Figura:** Representación gráfica de lo ocurrido en el 2012 cuando AlexNet ganó el primer lugar en el ILSVRC.

¿Por qué el entrenamiento supervisado de redes neuronales utilizando “backpropagation” no había sido tan exitoso antes? Geoffrey Hinton lo resumió en estos cuatro puntos:

- Nuestros conjuntos de datos etiquetados eran miles de veces más pequeños.
- Nuestros ordenadores eran millones de veces más lentos.
- Inicializábamos los pesos de forma estúpida.
- Utilizábamos el tipo de función de activación no lineal equivocada.

Entonces, después de esta larga historia, ¿qué es “deep learning”? Según IBM, el aprendizaje profundo es un subconjunto del área de aprendizaje automático (“machine learning”), que consiste esencialmente en una red neuronal con tres o más capas. Estas redes neuronales intentan simular el comportamiento del cerebro humano—aunque están lejos de igualar su capacidad—permitiéndole “aprender” de grandes cantidades de datos. O en otras palabras más coloquiales,

Deep Learning = Lots of training data + Parallel Computation + Scalable, smart algorithms.



**Figura:** La receta que resuelve muchos problemas en el área de “computer vision”. Imagen tomada de <https://www.computervisionblog.com/2015/05/deep-learning-vs-big-data-who-owns-what.html>.

# Redes Neuronales Convolucionales (CNN)

Una de las redes neuronales más usadas son las **redes neuronales convolucionales**, conocidas como CNN por sus siglas en inglés. Este tipo de arquitectura es bastante útil para datos de alta dimensionalidad y que están relacionados espacialmente (o temporalmente). Si utilizáramos una red neuronal totalmente conectada la cantidad de parámetros de la red sería innecesariamente grande.

## Redes Neuronales Convolucionales (CNN)

Por ejemplo, en los problemas de clasificación de imágenes, los píxeles vecinos suelen compartir información por su proximidad espacial, lo cual debería reflejarse en la arquitectura de la red. Basándose en esta observación, parece razonable tener redes neuronales que tengan campos receptivos locales en el sentido de que recojan información conjuntamente de entradas espacialmente cercanas.

# Redes Neuronales Convolucionales (CNN)

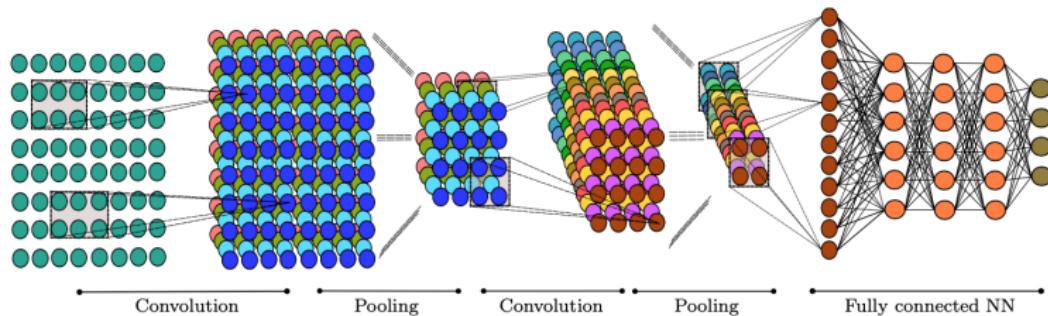
Además, en el procesamiento de imágenes, es relevante tener un clasificador que sea invariante bajo distintas operaciones como la traslación o la rotación de imágenes. Dado esto, es razonable codificar tales invariancias en la arquitectura.

Las redes convolucionales combinan los tres aspectos ya mencionados: manejo de entradas de alta dimensionalidad, procesamiento de datos relacionados espacialmente de manera local, y ser “invariantes” a traslaciones.

# Redes Neuronales Convolucionales (CNN)

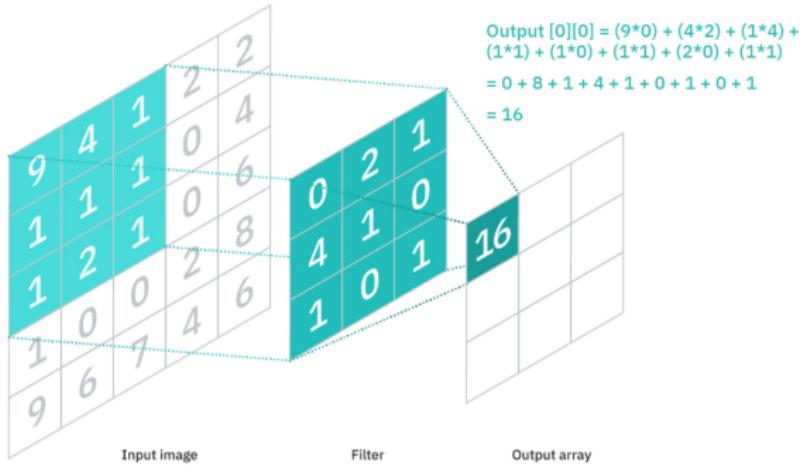
Vale la pena decir que las CNN realmente no son invariantes a traslaciones, sino **equivariantes**. Por otro lado, las convoluciones no son realmente la operación de **convolución** que se define para funciones continuas y discretas, estas son realmente **correlaciones cruzadas**.

# Redes Neuronales Convolucionales (CNN)



**Figura:** Arquitectura de una red neuronal convolucional con bloques convolucionales bidimensionales y submuestreo  $2 \times 2$  como operación de agrupación. Imagen tomada de [4].

# Redes Neuronales Convolucionales (CNN)



**Figura:** En la figura se puede apreciar como se realiza la “convolución” con un filtro de tamaño  $3 \times 3$ . El filtro recorre toda la imagen y cada valor calculado se almacena en un arreglo. Estos filtros, los cuales se encuentran en cada capa convolucional, le ayudan a la red a aprender características relevantes de las imágenes. Imagen tomada de <https://www.ibm.com/cloud/learn/convolutional-neural-networks>.

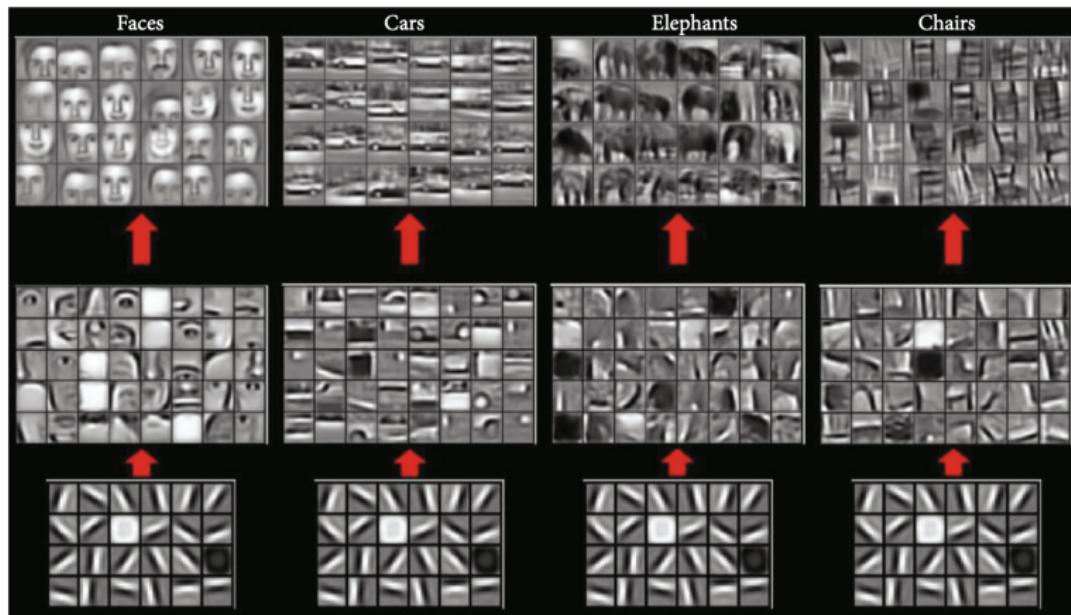
# Redes Neuronales Convolucionales (CNN)

Las capas de “**pooling**” realizan una reducción de la dimensionalidad por medio de un submuestreo. Hay dos tipos principales de “pooling”:

- “**Max pooling**”: A medida que el filtro se desplaza por la entrada, selecciona el píxel con el valor máximo para enviarlo a la matriz de salida.
- “**Pooling promedio**”: En este caso se calcula el valor medio dentro del campo receptivo para enviarlo a la matriz de salida.

Aunque se pierde mucha información en la capa de pooling, esta técnica ayuda a reducir la complejidad de la red, lo cual limita el riesgo de sobreajuste.

# Redes Neuronales Convolucionales (CNN)



**Figura:** Para las diferentes capas convolucionales de una CNN, las características de la imagen aprendidas por cada capa son diferentes. Los patrones aprendidos por las capas más superficiales tienden a ser más generales, mientras que para capas más profundas, las características aprendidas son más específicas dependiendo del conjunto de datos con el que se esté trabajando. Imagen tomada de [5].

## Capas Convolucionales

En las capas convolucionales tenemos un conjunto de filtros que en el entrenamiento de la red aprenden a detectar patrones interesantes de las imágenes del conjunto de entrenamiento. Estos filtros se pueden entender como matrices, normalmente cuadradas, de dimensionalidad impar. Como ya se había comentado, con estos filtros no se hace realmente una convolución, sino una correlación.

Supongamos que tenemos una imagen de tamaño  $n_x \times n_y$  y un filtro de tamaño  $f \times f$ . ¿De qué tamaño es el resultado de la “convolución” entre la imagen y el filtro? El resultado es el siguiente:

$$(n_x - f + 1) \times (n_y - f + 1).$$

Nótese que el resultado de la convolución nos entrega un resultado que no es del tamaño de la imagen de entrada. Si queremos que el resultado de la convolución tenga la misma dimensión de la imagen de entrada, u otras dimensiones, debemos hacerle “padding” a la imagen de entrada, lo cual consiste en añadir píxeles adicionales a las fronteras de la imagen.

Si el “padding” es  $p_x$  y  $p_y$ , y tenemos un filtro de tamaño  $f \times f$ , las dimensiones de la convolución de una imagen con “padding” son las siguientes:

$$(n_x + 2p_x - f + 1) \times (n_y + 2p_y - f + 1).$$

¿A qué deberían ser iguales  $p_x$  y  $p_y$  para que el resultado de la convolución tenga las mismas dimensiones de la imagen de entrada?

## Capas Convolucionales

Otro parámetro importante de las capas convolucionales es el “stride”: el desplazamiento que realiza el filtro en cada eje cada vez que se calcula un valor de la convolución.

Si tenemos que el “stride” es  $s_x$  y  $s_y$ , el resultado de hacer una convolución con un filtro de tamaño  $f \times f$  tiene estas dimensiones:

$$\left\lfloor \frac{n_x - f}{s_x} + 1 \right\rfloor \times \left\lfloor \frac{n_y - f}{s_y} + 1 \right\rfloor.$$

Si incluimos “padding” tenemos lo siguiente:

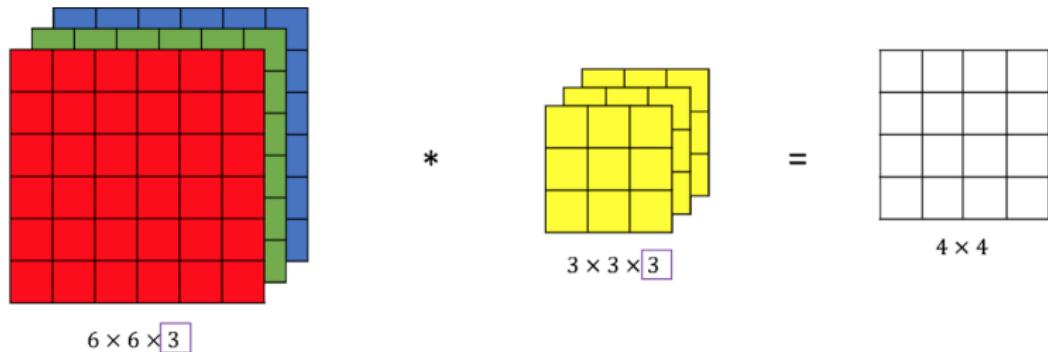
$$\left\lfloor \frac{n_x + 2p_x - f}{s_x} + 1 \right\rfloor \times \left\lfloor \frac{n_y + 2p_y - f}{s_y} + 1 \right\rfloor.$$

## Capas Convolucionales

Solo las imágenes en blanco y negro son arreglos bidimensionales, en cambio, las imágenes a color son arreglos tridimensionales. Si se trabaja con el formato RGB, tenemos un arreglo bidimensional, o matriz, por cada canal, en este caso tres canales en total: uno para el rojo (R), otro para el verde (G), y uno para el azul (B).

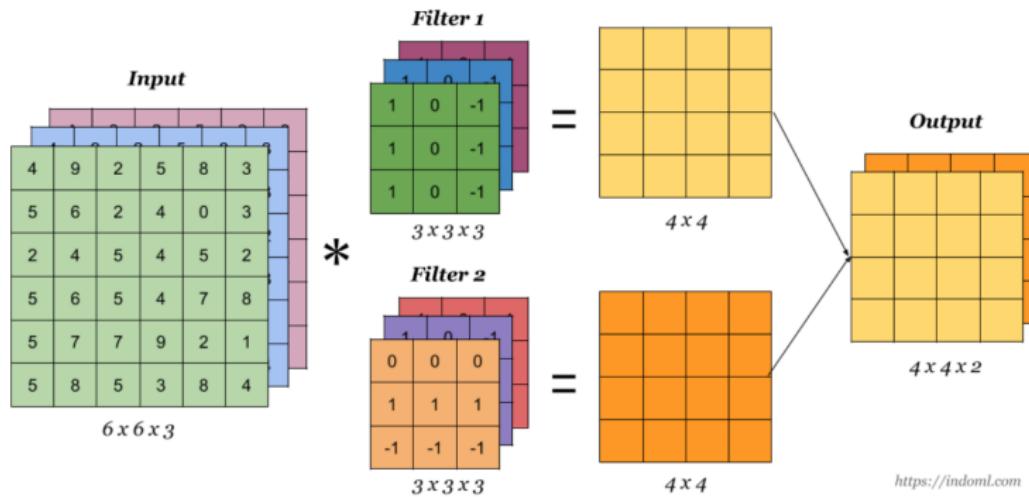
Si queremos hacer convolución con volúmenes, los filtros de las capas convolucionales también deben ser volúmetricos, es decir, la dimensionalidad de la tercera dimensión debe coincidir con la dimensionalidad de la imagen de entrada.

## Capas Convolucionales



**Figura:** Convolución de una imagen RGB de dimensiones  $6 \times 6 \times 3$  con un filtro de tamaño  $3 \times 3 \times 3$ . Nótese que el resultado es un arreglo de tamaño  $4 \times 4 \times 1$ . Imagen tomada de <https://towardsdatascience.com/beginners-guide-to-understanding-convolutional-neural-networks-ae9ed58bb17d>.

# Capas Convolucionales



**Figura:** Convolución de una imagen RGB de dimensiones  $6 \times 6 \times 3$  con dos filtros de tamaño  $3 \times 3 \times 3$ . Nótese que el resultado es un arreglo de tamaño  $4 \times 4 \times 2$  y que ambos filtros tiene el mismo tamaño. Imagen tomada de <https://towardsdatascience.com/beginners-guide-to-understanding-convolutional-neural-networks-ae9ed58bb17d>.

## Capas Convolucionales

En general, el resultado entregado por una capa convolucional será igual a lo siguiente:

$$\left\lfloor \frac{n_x + 2p_x - f}{s_x} + 1 \right\rfloor \times \left\lfloor \frac{n_y + 2p_y - f}{s_y} + 1 \right\rfloor \times n_f,$$

donde  $n_f$  es el número de filtros.

Una vez hecha la convolución, se le añade un “bias” y una función no lineal, normalmente la ReLU, se evalúa en cada elemento de esta última operación, acción que no cambia las dimensiones de la “convolución”:

$$\phi(F * X + b).$$

Por cierto, los parámetros del los filtros de las capas convolucionales no están dados, se aprenden durante el entrenamiento de la red.

## Capas Convolucionales

Sean  $f^{(l)}$ ,  $p^{(l)}$ ,  $s^{(l)}$  y  $n_f^{(l)}$  las dimensiones del filtro, el “padding”, el “stride” y el número de filtros, respectivamente, de la capa  $l$ . Entonces, si las dimensiones de la entrada de una capa convolucional están dadas por

$$n_x^{(l-1)} \times n_y^{(l-1)} \times n_f^{(l-1)},$$

las dimensiones de la salida de la capa convolucional son las siguientes:

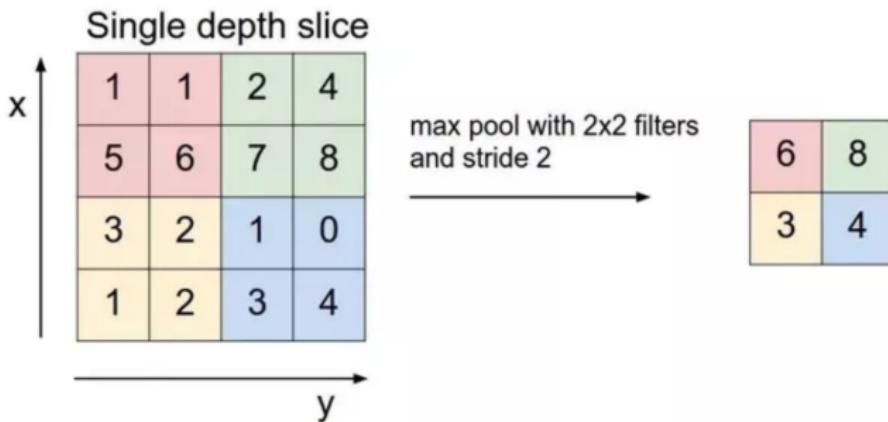
$$\left\lfloor \frac{n_x + 2p_x - f}{s_x} + 1 \right\rfloor \times \left\lfloor \frac{n_y + 2p_y - f}{s_y} + 1 \right\rfloor \times n_f^{(l)}.$$

## Pooling

Otro de los elementos importantes de una CNN son las capas de “pooling”: su objetivo principal es reducir la dimensionalidad de la información entregada por una capa anterior procurando mantener las características más relevantes.

Como mencionamos anteriormente, el “pooling” se puede llevar a cabo de distintas manera, siendo “max pooling” el más común. Esta operación también posee algunos de los hiperparámetros de las capas convolucionales, en particular, el tamaño del “filtro” y el “stride”.

## Pooling



**Figura:** “Max pooling” de una imagen de dimensiones  $4 \times 4$  con un filtro de tamaño  $2 \times 2$  y un “stride” igual a dos. Nótese que el resultado es un arreglo de tamaño  $2 \times 2$ . Imagen tomada de <https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>.

Es importante mencionar que una vez la información que entra a la red pasa por las capas convolucionales y de “pooling”, la información se “vectoriza” en una capa conocida como “flatten”. Esto se hace con el fin de entregar la información en un formato que una red neuronal totalmente conectada pueda procesar.

Supongamos que la entrada de una red tiene dimensiones  $32 \times 32 \times 3$  y que las capas convolucionales y de “pooling” tienen los siguientes hiperparámetros:

- Capa convolucional:  $f^{(1)} = 5$ ,  $s_x^{(1)} = s_y^{(1)} = 2$ ,  $n_f^{(1)} = 6$ .
- “Pooling”:  $f^{(1)} = 2$ ,  $s_x^{(1)} = s_y^{(1)} = 2$ .
- Capa convolucional:  $f^{(2)} = 5$ ,  $s_x^{(2)} = s_y^{(2)} = 2$ ,  $n_f^{(2)} = 6$ .
- “Pooling”:  $f^{(2)} = 2$ ,  $s_x^{(2)} = s_y^{(2)} = 2$ .

¿Cuáles son las dimensiones que se manejan en cada etapa de la red? ¿Cuál es la dimensión de la capa de entrada de la red neuronal totalmente conectada? ¿Cuántos parámetros tiene la sección convolucional de la CNN?

# BIBLIOGRAFÍA

- 1 Dangeti, P. "*Statistics for machine learning,*" Packt Publishing Ltd., 2017.
- 2 Haykin, S., "*Neural networks and learning machines*", Pearson Education India, 2010.
- 3 Stuart R., Norvig P, "*Artificial intelligence: a modern approach,*" 2002.
- 4 J. Berner et al., "*The modern mathematics of deep learning,*" arXiv preprint arXiv:2105.04026, 2021.
- 5 Y. Wang et al., "*Multiclassification of endoscopic colonoscopy images based on deep transfer learning*", Computational and Mathematical Methods in Medicine, 2021.
- 6 Andrey Kurenkov, "*A Brief History of Neural Nets and Deep Learning,*" Skynet Today, 2020.