uc3m | Universidad Carlos III de Madrid

Máquinas Discriminativas Profundas (*Deep Learning*)

Ejemplo práctico en laboratorio

http://www.tsc.uc3m.es/~mlazaro/Docencia/DL.html

Marcelino Lázaro

Universidad Carlos III de Madrid

Instrucciones iniciales

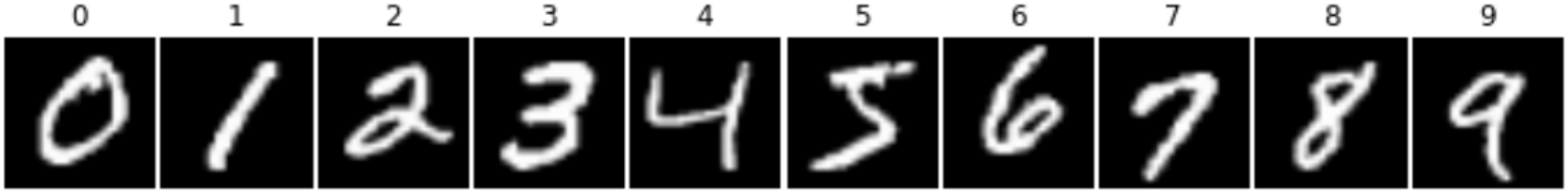
- Ficheros necesarios en el directorio de trabajo

  `http://www.tsc.uc3m.es/~mlazaro/Docencia/DL.html`

  - Python Notebook
    - ★ *practicaDL-2021.ipynb*
  - Fichero de métodos de aprendizaje
    - ★ *metodosDL.py*
  - Ficheros de datos (`http://yann.lecun.com/exdb/mnist/`)
    - ★ *train-images-idx3-ubyte.gz*
    - ★ *train-labels-idx1-ubyte.gz*
    - ★ *t10k-images-idx3-ubyte.gz*
    - ★ *t10k-labels-idx1-ubyte.gz*
- Librerías necesarias
  - *numpy*
  - *matplotlib*
  - *gzip*
  - *tensorflow*
  - *scikit-learn*

Índice de contenidos

- Aprendizaje profundo (*"deep learning"*) para clasificación



  - Aplicación a una versión reducida de la base de datos MNIST
- Breve descripción de las funciones de aprendizaje disponibles
- Diseño de varias soluciones neuronales
  - Diseño de una red neuronal no profunda (1 capa oculta)
  - Diseño de una red neuronal profunda (4 capas oculta)
  - Diseño de una red de autocodificadores apilados (*SDAs: Stacked Denoising Autoencoders*)
    - ★ Entreno de autocodificadores
    - ★ Uso de autocodificadores para la construcción de una red profunda
  - Redes profundas con Dropout (y/o Batch normalization)
  - Redes convolucionales (*CNNs: Convolutional Neural Networks*)

Python

Descripción de las funciones presentes en *metodosDL.py*

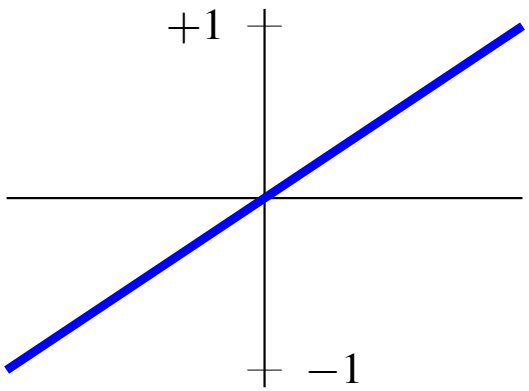
Funciones en Python (evaluación MLPs)

- (o,H)=mlp(x,W,tAct)
  - x: patrones de entrada (Np $\times$ Ne)
  - W: pesos de la red: lista the No + 1 elementos
    - Capa oculta 1 : (Ne+1) $\times$ Nn1
    - Capa oculta i : (Nn(i-1)+1) $\times$ Nni
    - Capa de salida : (Nnu+1) $\times$ Ns
  - tAct : parámetro que define el tipo de funciones de activación de las diferentes capas (lista de No+1 elementos)
    - 'linear'
    - 'sigmoid'
    - 'tanh'
    - 'relu'
    - 'softmax'

  Nni: número de neuronas de la capa oculta i (si i=0, Nni=Ne)
  Nnu: número de neuronas de la última capa oculta
  No: número de capas ocultas
  - o: salida de la red neuronal (Np $\times$ Ns)
  - H: lista con las salidas de las capas ocultas (No elementos)

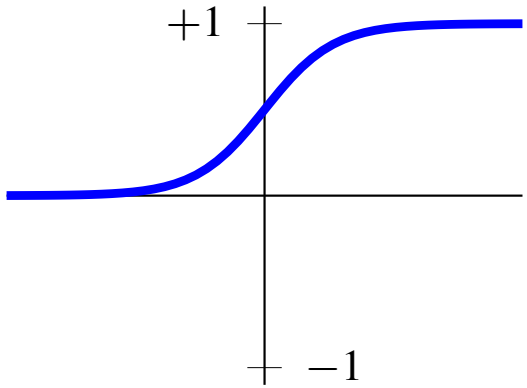
Funciones de activación

$f(x) = x$



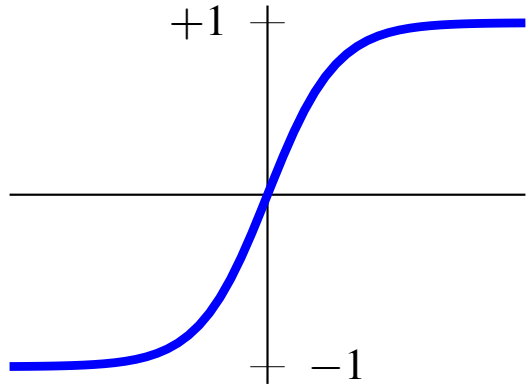
Lineal

$f(x) = \frac{1}{1+e^{-x}}$



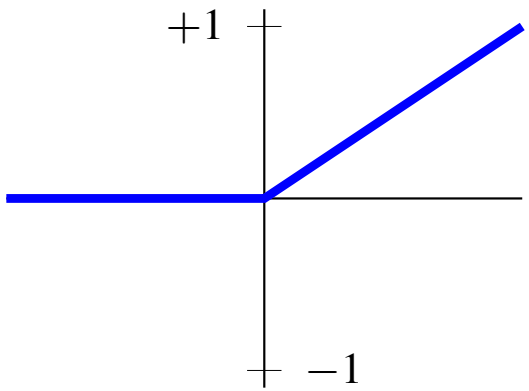
Logística (sigmoide)

$f(x) = \tanh(x)$



Tangente Hiperbólica

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$



Lineal rectificada (ReLU)

$$o_i = \frac{e^{z_i}}{\sum_{k=1}^{M} e^{z_k}}$$

Softmax

Funciones en Python (entrenamiento MLPs)

Entrenamiento de MLPs con distinto número de capas ocultas

- (W,paso,coste,dWm)=entrena_mlp(x,y,W,Nepoch)
  - x: patrones de entrada (Np $\times$ Ne)
  - y: etiquetas de salida (Np $\times$ Nc)
  - W: lista con los pesos de la red (No + 1 elementos)
  - Nepoch: lista con el número de iteraciones y el tamaño del mini-batch
    - ★ Nepoch=[Niter,Nbatch]
- Parámetros adicionales
  - fCoste: función de coste a minimizar ('mmse', 'entropia', 'wmmse')
  - optimizador: método de optimización ('gradiente', 'momento')
  - tAct: tipo de funciones de activación de las capas de la red
  - pDO: probabilidad de Drop-Out de la entrada y capas ocultas (lista No+1 el.)
  - paso: parámetros del paso de adaptación
    - ★ Gradiente: paso=[mu,muCrec,muDec]
    - ★ Momento: paso=[mu,momento]
  - flagEvo: True para que se obtenga el coste cada iteración

Nc : número de clases para un problema de clasificación (ONE-HOT ENCODING)

Funciones en Python (entrenamiento AE)

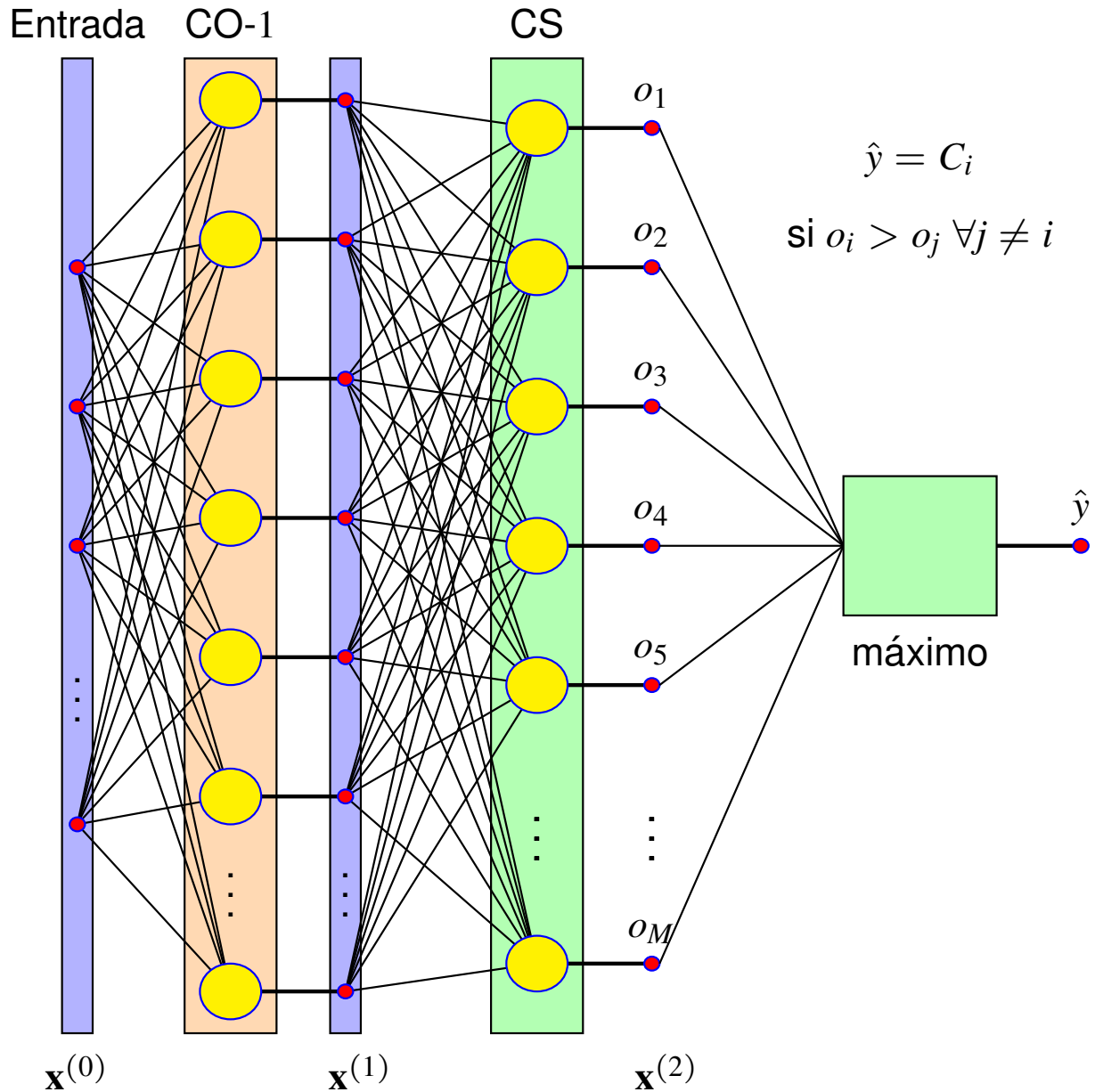
Funciones para el entrenamiento de autocodificadores (AE)

- (we,wd,coste,mu)=entrena_AE(x,we,wd,Nepoch,tAct,mu)
  Entrenamiento de un autocodificador
- (we,wd,coste,mu)=entrena_AEtied(x,we,wd,Nepoch,tAct,mu)
  Entrenamiento de un autocodificador con pesos “atados”
- (we,wd,coste,mu)=entrena_DAE(x,we,wd,Nepoch,tAct,mu,pRuido)
  Entrenamiento de un autocodificador con ruido
- (we,wd,coste,mu)=entrena_DAEtied(x,we,wd,Nepoch,tAct,mu,pRuido)
  Entrenamiento de un autocodificador con ruido y pesos “atados”
  - we: pesos del codificador (Ne+1 $\times$ Nn)
  - wd: pesos del decodificador (Nn+1 $\times$ Ne)
  - pRuido: probabilidad de ruido (de poner una entrada a cero)

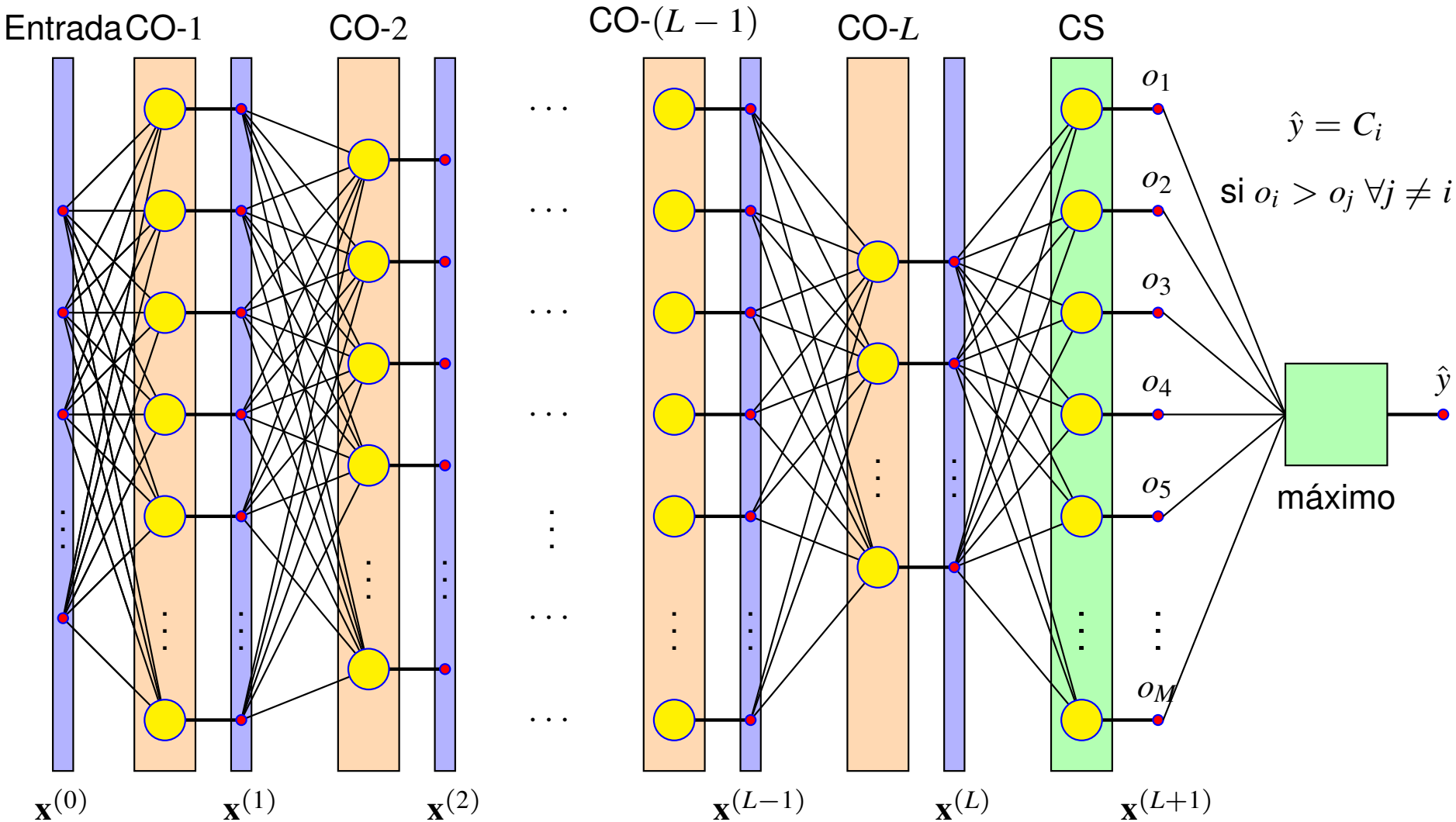
Perceptrón multicapa

(MLP: *Multi-Layer Perceptron*)

Red neuronal no profunda: 1 capa oculta (M-ário)



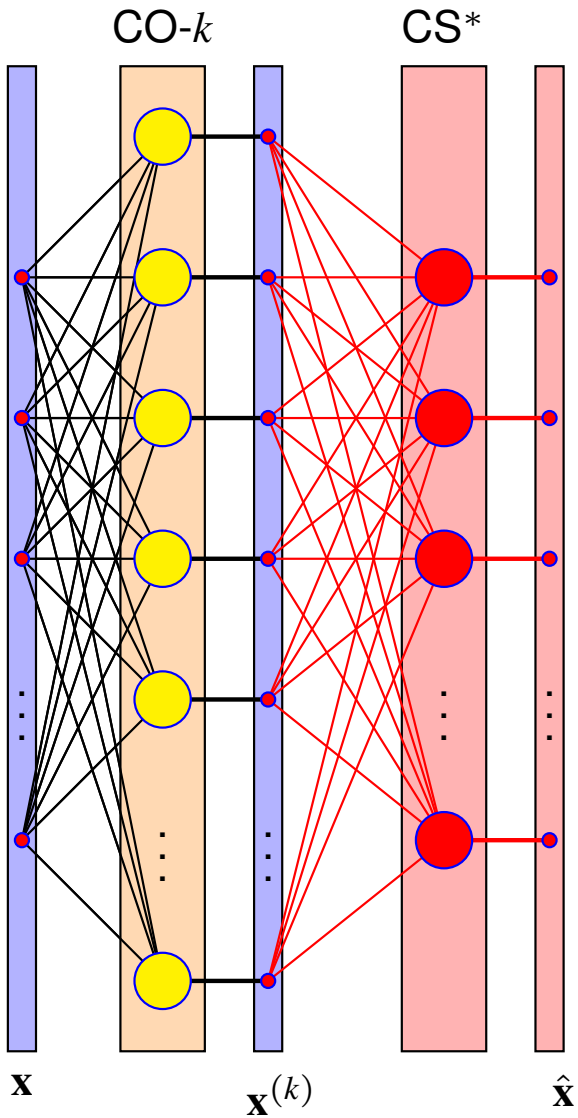
Red profunda: red neuronal con L capas ocultas (M-ário)



Autocodificadores Apilados

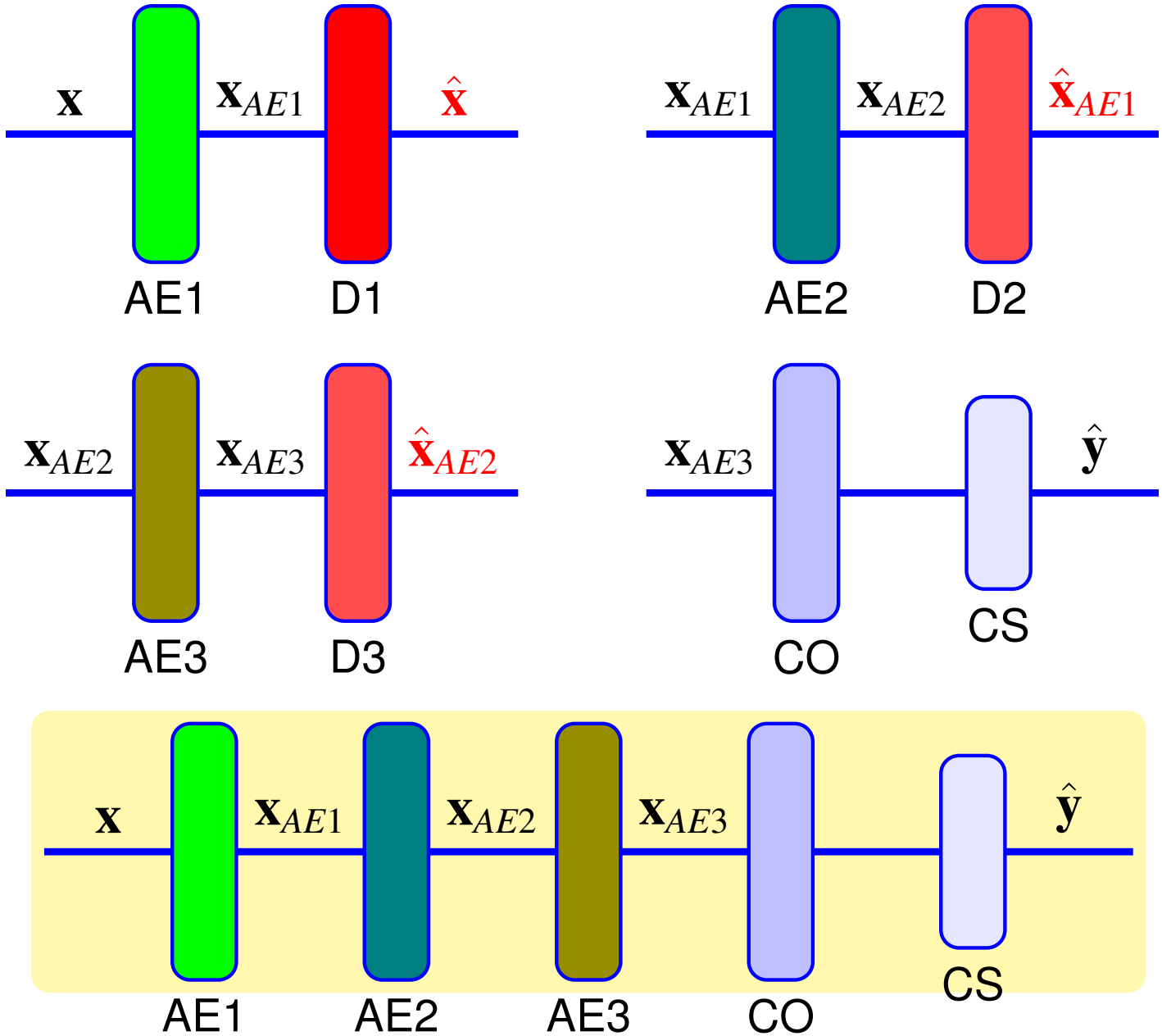
(SDAs: *Stacked Denoising Autoencoders*)

Autocodificadores (*"autoencoders"*)

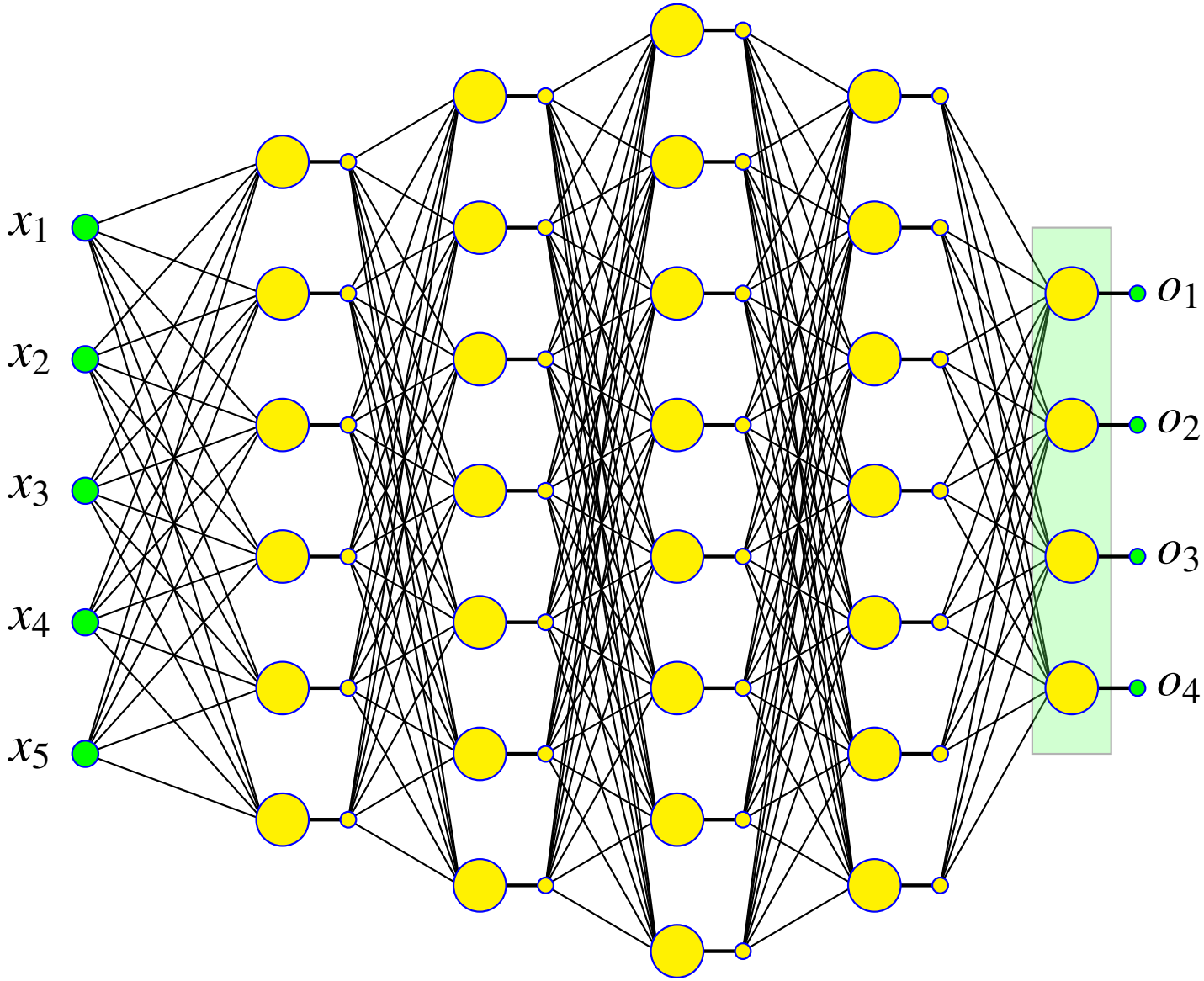


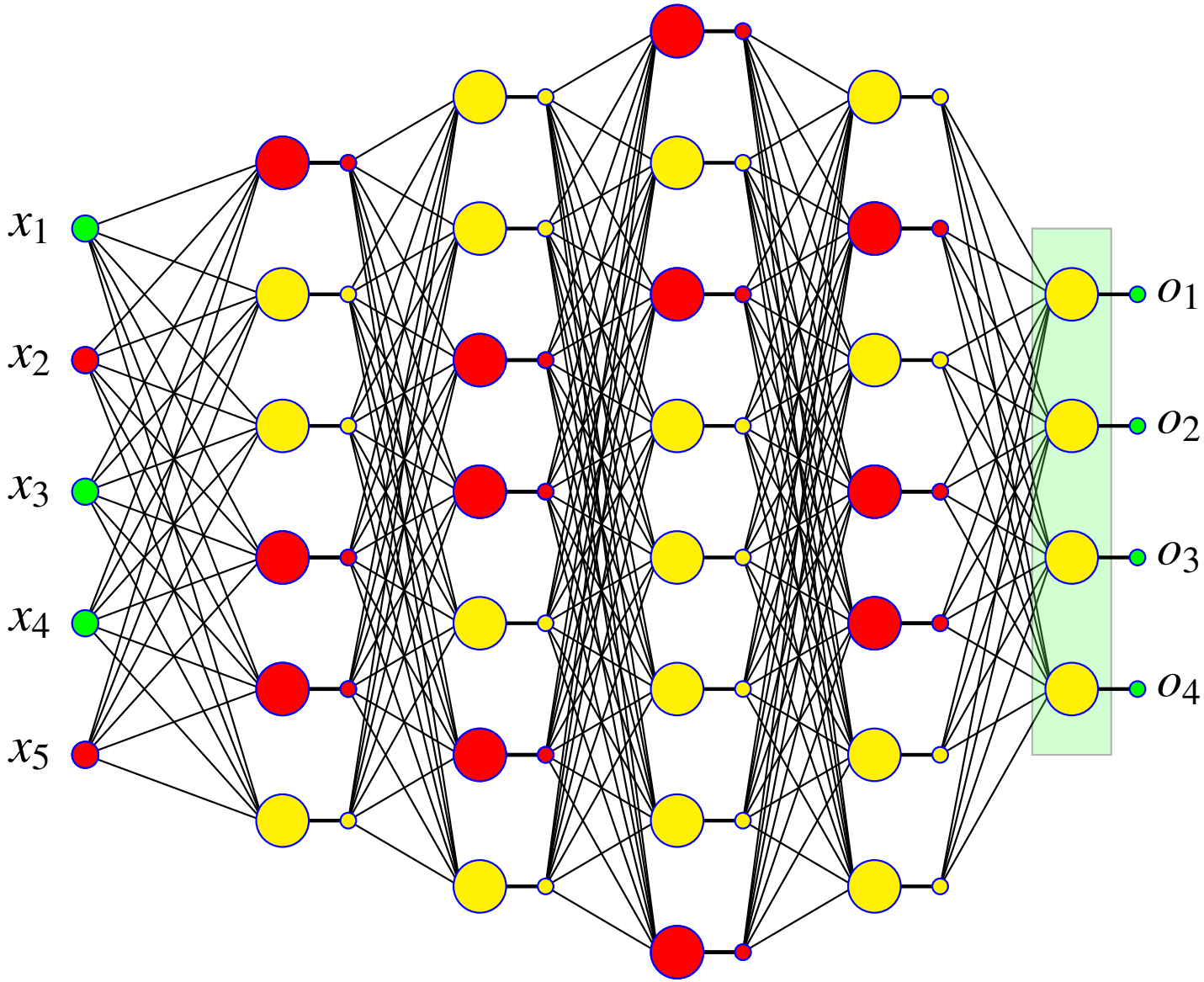
- El entrenamiento de una capa oculta se realiza para obtener una proyección de su entrada sin pérdida de información
  - Entrada: $\mathbf{x} = \mathbf{x}^{(k-1)}$
  - Salida deseada: $\hat{\mathbf{x}} = \mathbf{x}^{(k-1)}$
- Denoising autoencoders
  - Se incluye ruido en la entrada para hacer la proyección robusta a perturbaciones sobre la entrada (y para evitar soluciones triviales)
    - ★ Entrada: $\mathbf{x} = \mathbf{x}^{(k-1)} + \mathbf{n}$
    - ★ Salida deseada: $\hat{\mathbf{x}} = \mathbf{x}^{(k-1)}$
- La capa de salida se descarta
  - Se usa como mecanísmo para evitar la pérdida de información, al garantizar que la proyección obtenida es reversible

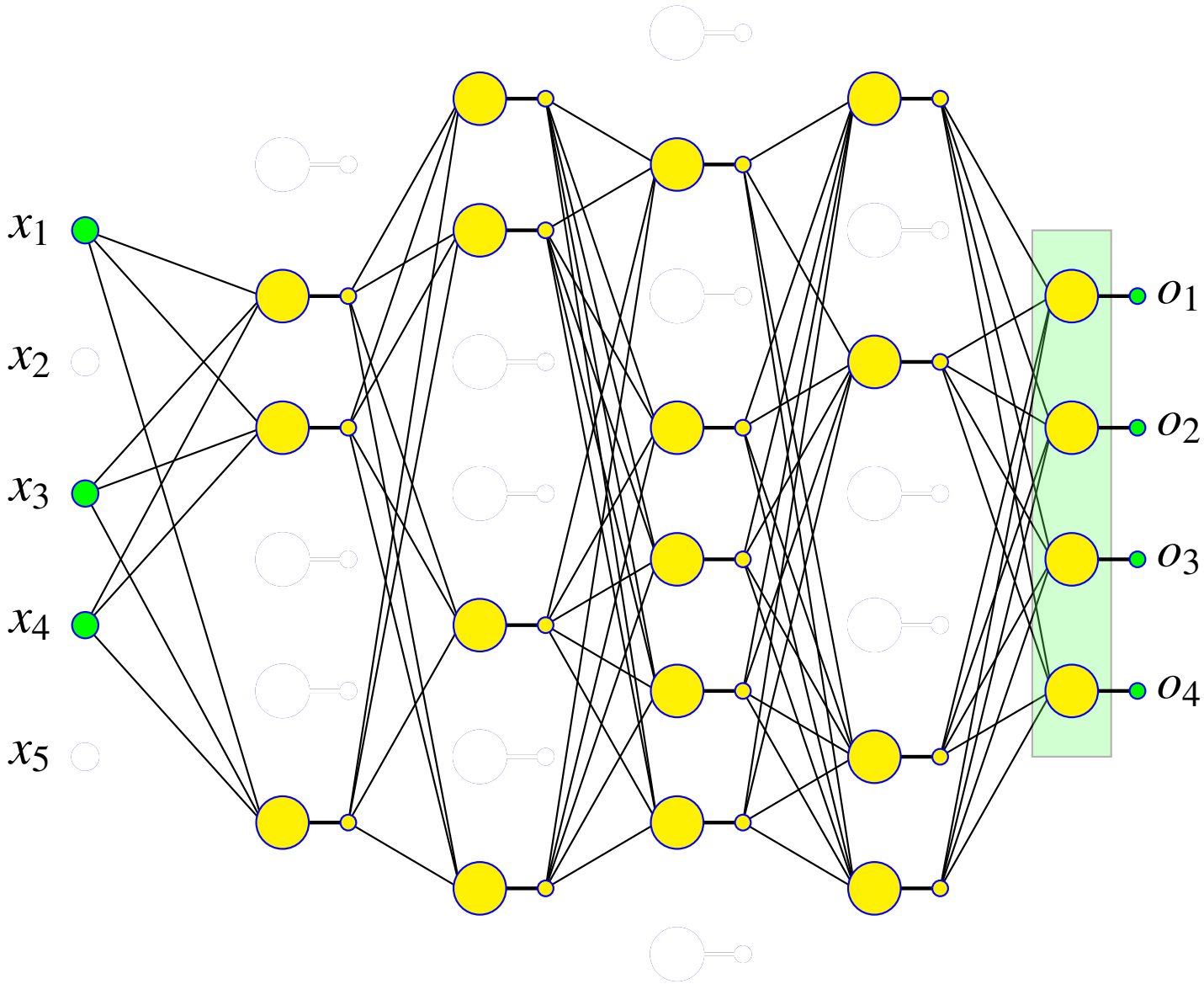
Construcción de red profunda con autocodificadores

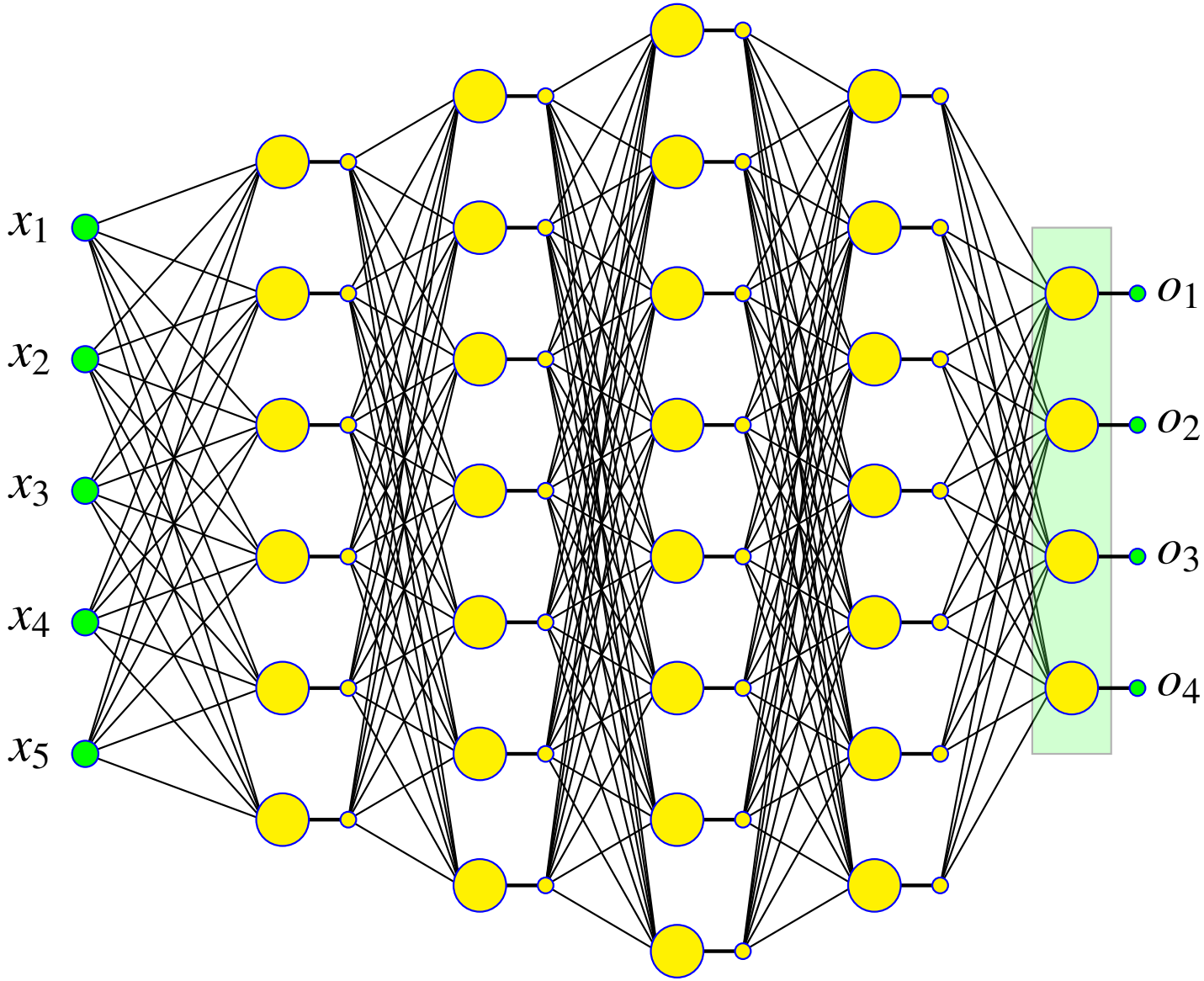


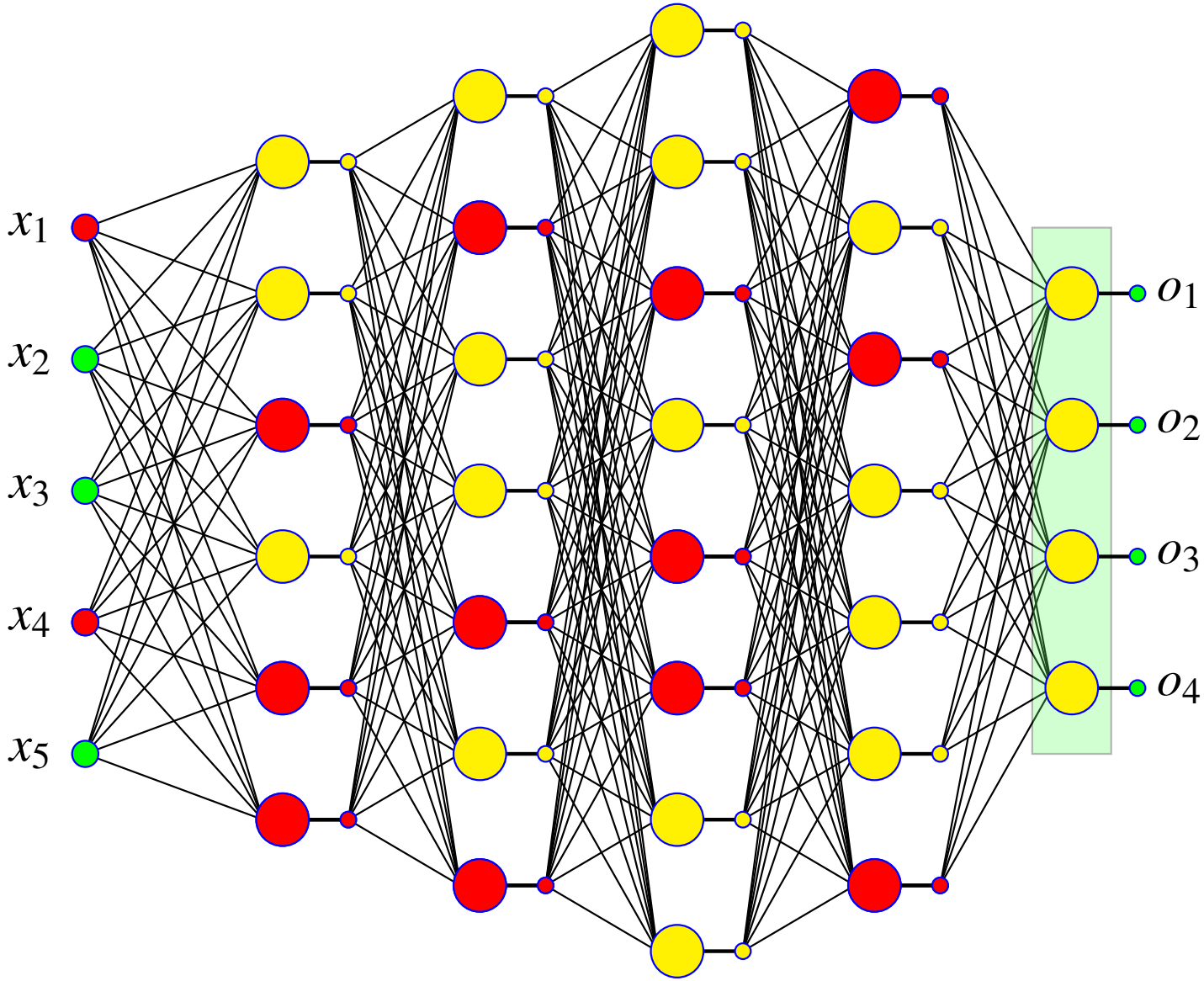
Redes profundas con Drop-Out

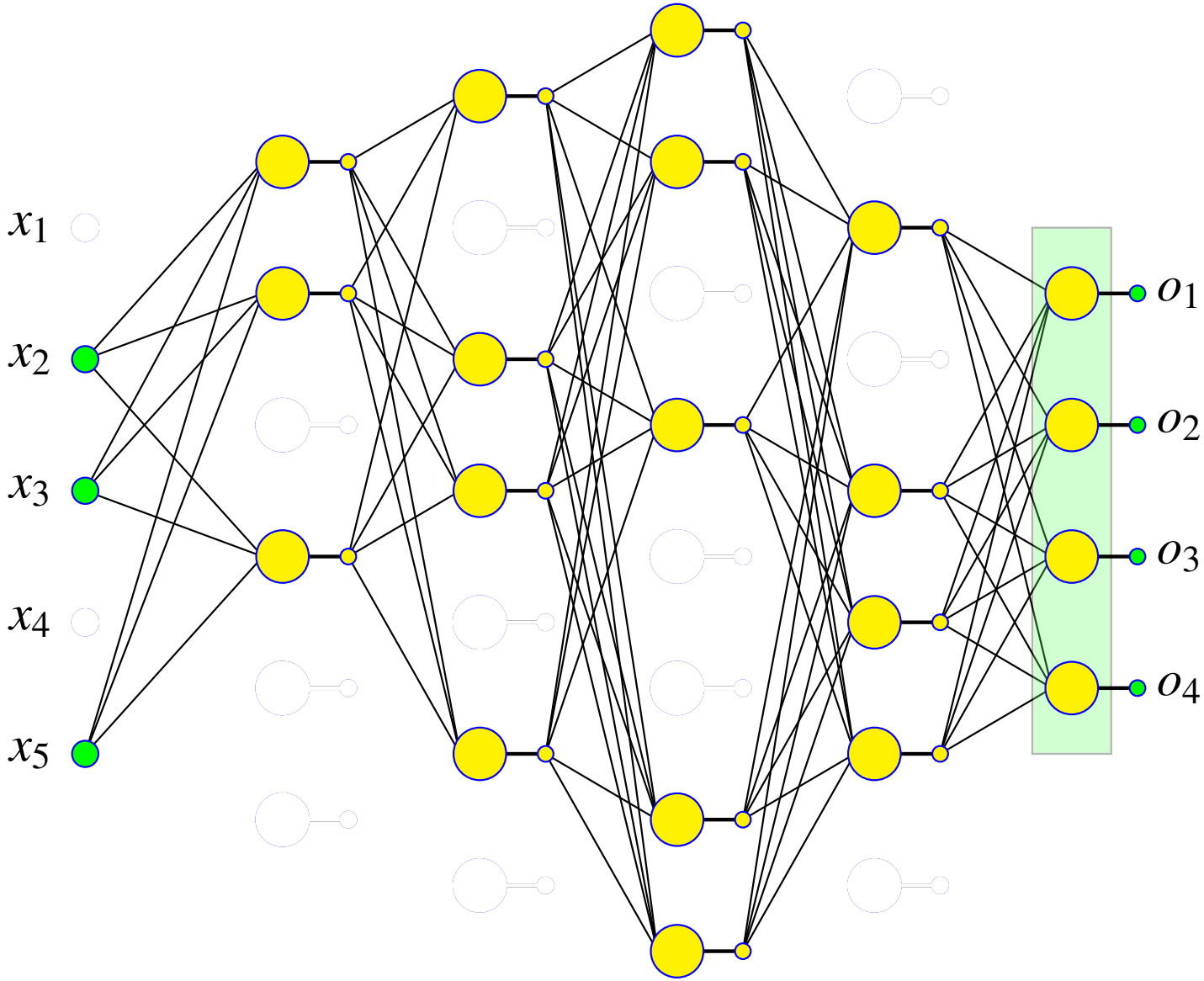

x_1
x_2
x_3
x_4
x_5
o_1
o_2
o_3
o_4


x_1
x_2
x_3
x_4
x_5
o_1
o_2
o_3
o_4



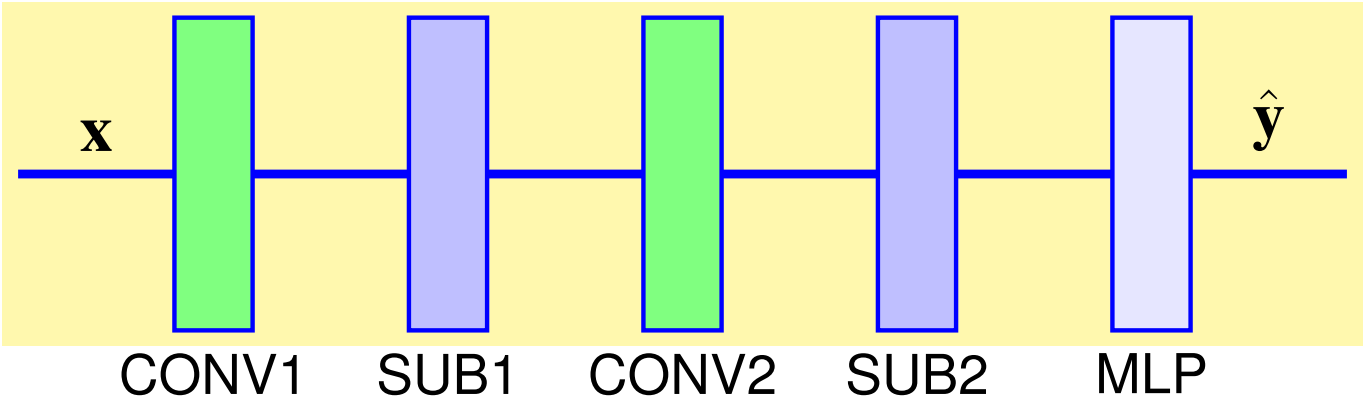


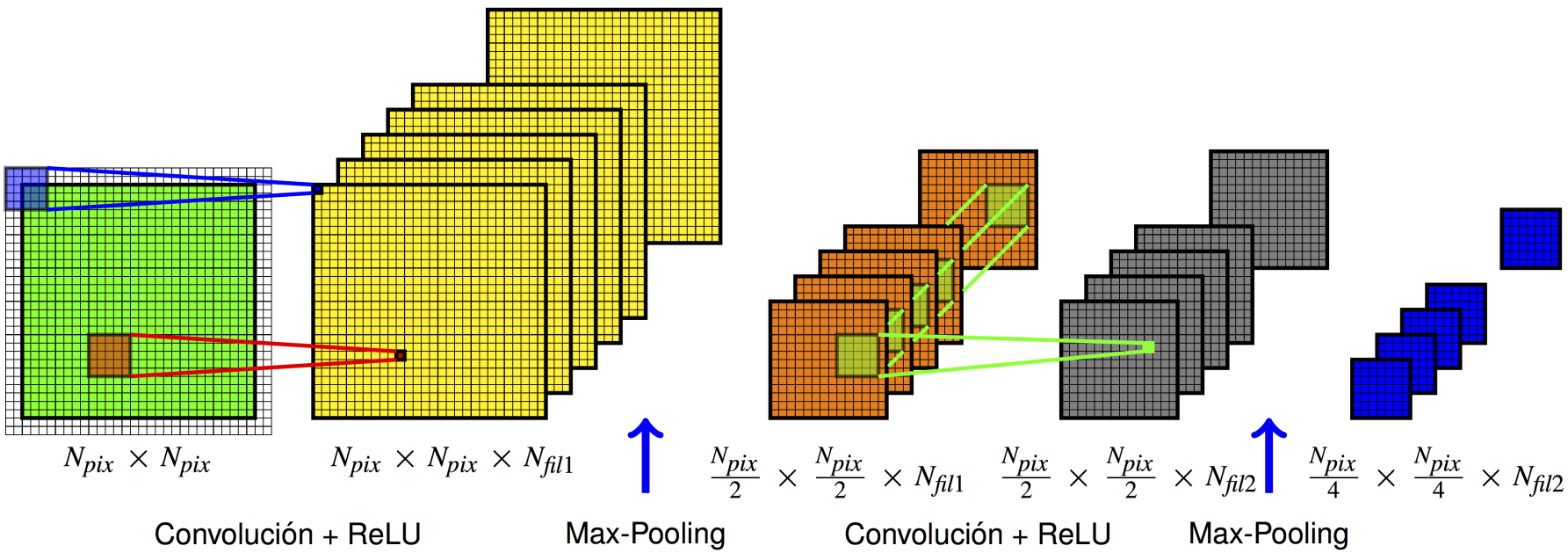




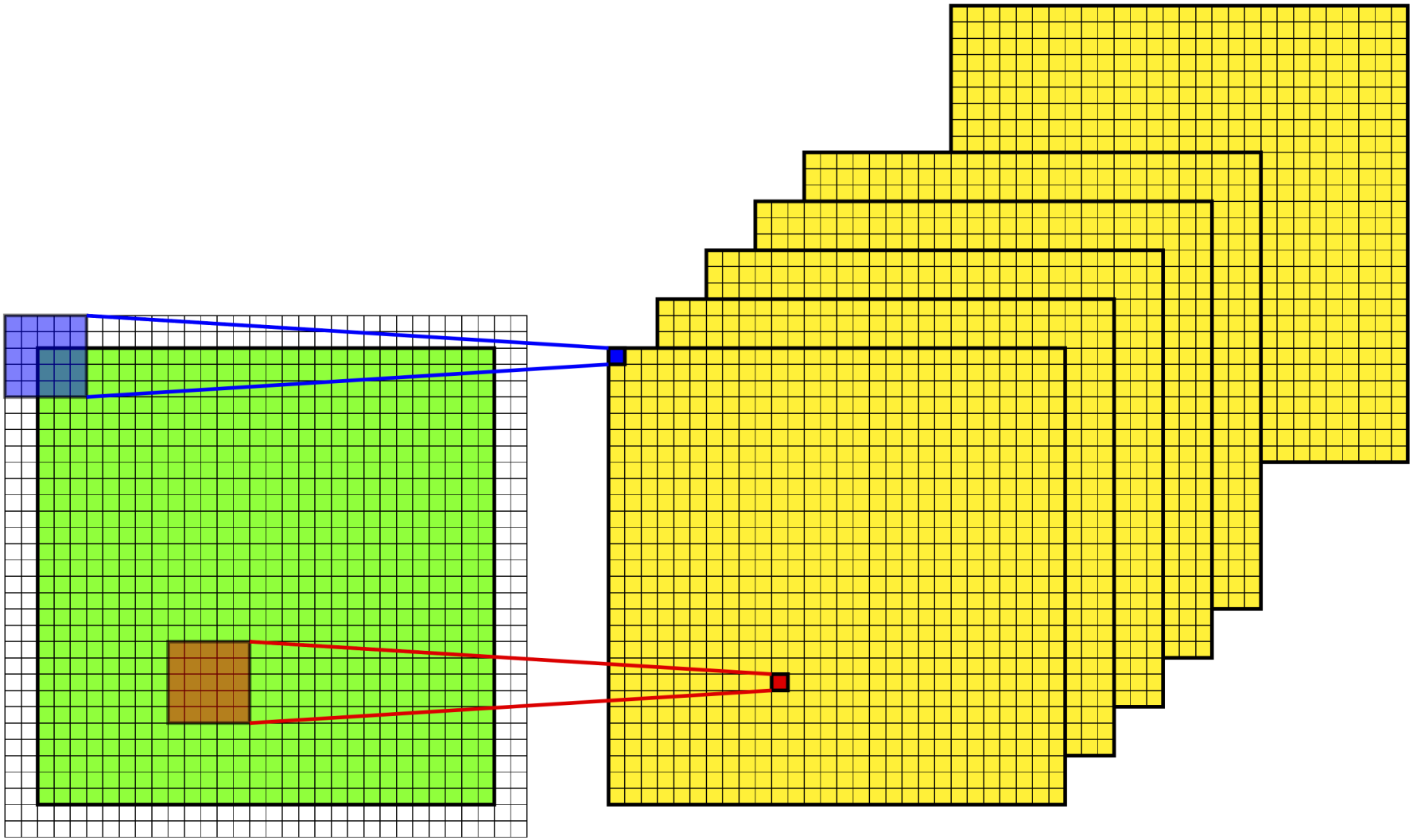
Redes Convolucionales

Convolutional Neural Network (CNN)





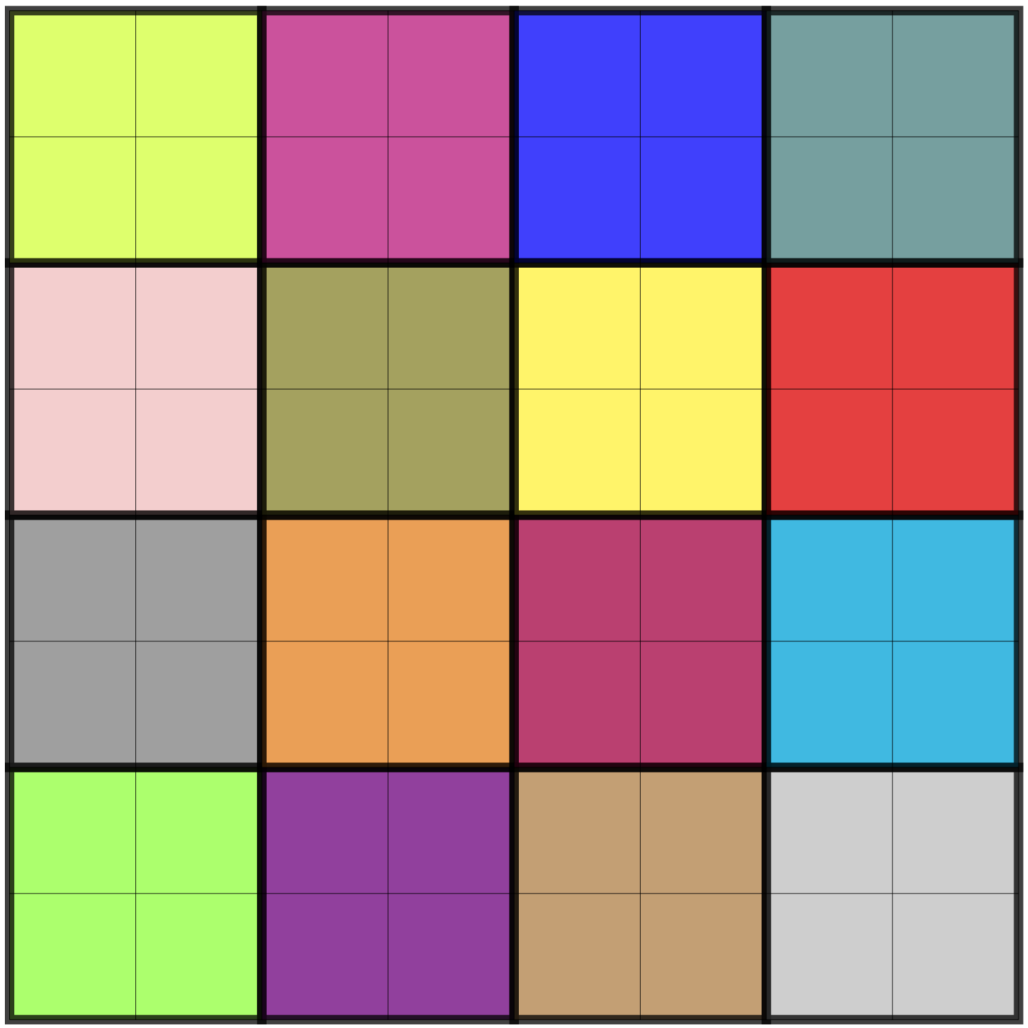
Convolución (con "*padding*")

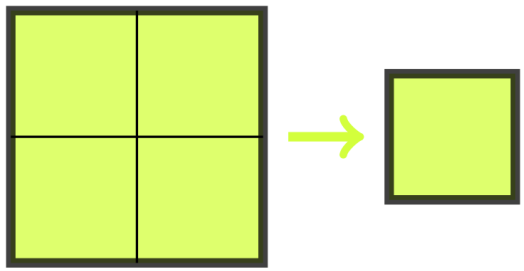


$N_{pix} \times N_{pix}$

$N_{pix} \times N_{pix} \times N_{fil}$

Max-Pooling





Maximum Value

Resultados (Aproximados)

Resultados aproximados esperables

- Resolución 10×10 (2.000 patrones de entrenamiento)
  - Red neuronal de 1 capa oculta: $\approx 87\,\%$
  - Red neuronal de 4 capas ocultas (*tanh*): $\approx 86\,\%$
  - Red neuronal profunda con autocodificadores: $\approx 88{,}5\,\%$
  - Red neuronal profunda (*relu*) con Dropout: $\approx 89{,}75\,\%$
  - Red neuronal profunda (*relu*) con Dropout + BN: $\approx 90{,}5\,\%$
  - Red CNN: $\approx 91{,}5\,\%$
- Resolución 28×28
  - Red neuronal de 1 capa oculta: $\approx 92\,\%$
  - Red neuronal profunda con autocodificadores: $\approx 98\,\%$
  - Red neuronal profunda con Dropout: $\approx 98{,}25\,\%$
  - Red CNN: $\approx 99\,\%$
  - Resultados detallados (a lo largo del tiempo)
    - ★ `http://yann.lecun.com/exdb/mnist/`