

UNIVERSIDADE DA CORUÑA

# APRENDIZAJE AUTOMÁTICO

TEMA 5:  
OTROS CONCEPTOS

# DEEP LEARNING

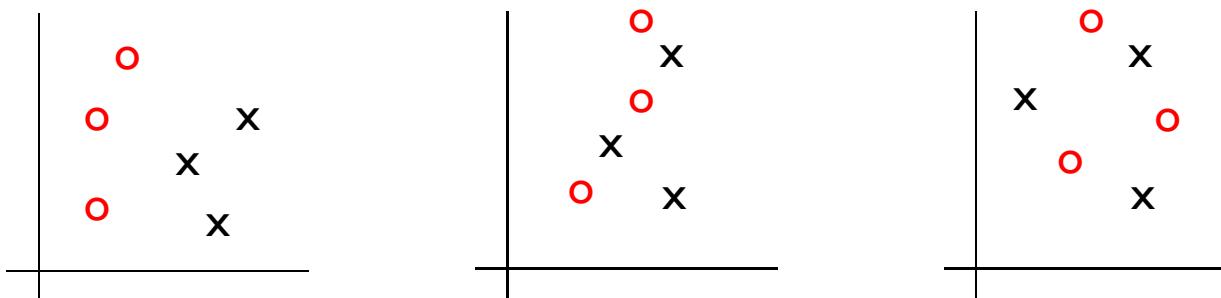
- Modelo tradicional de reconocimiento de patrones:
  - Desde finales de los años 50



- Características fijas + clasificador
- La extracción de características requiere un gran esfuerzo por parte de los expertos
- Este modelo no ha evolucionado mucho

# DEEP LEARNING

- **Modelo tradicional de reconocimiento de patrones:**
  - La extracción manual de características es una etapa crucial en el proceso de clasificación
    - Distintos conjuntos de características pueden hacer que se tengan patrones de las siguientes maneras:



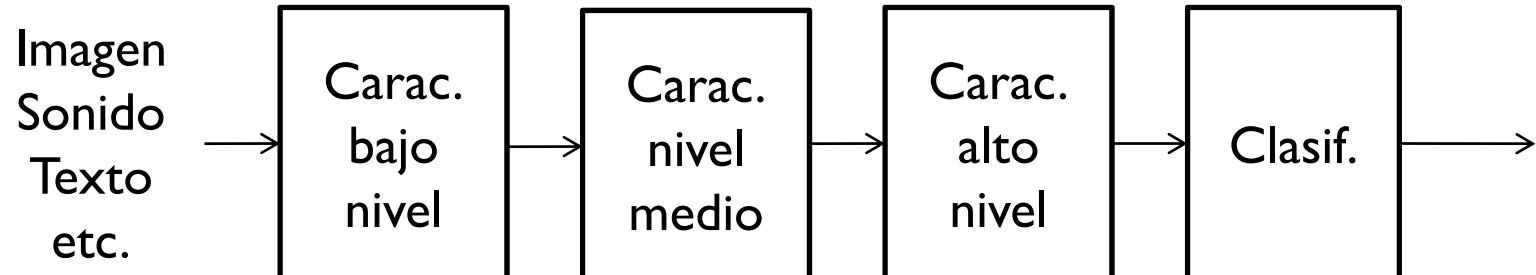
- El clasificador tendrá más o menos éxito según estas características
- ¿Es posible desarrollar un sistema que extraiga las características de forma automática?

# DEEP LEARNING

- *End-to-end learning / Feature learning / Deep learning:*



- Se dice que es profundo (*deep*) si tiene más de una etapa de extracción de características:
  - Extracción de características jerárquica



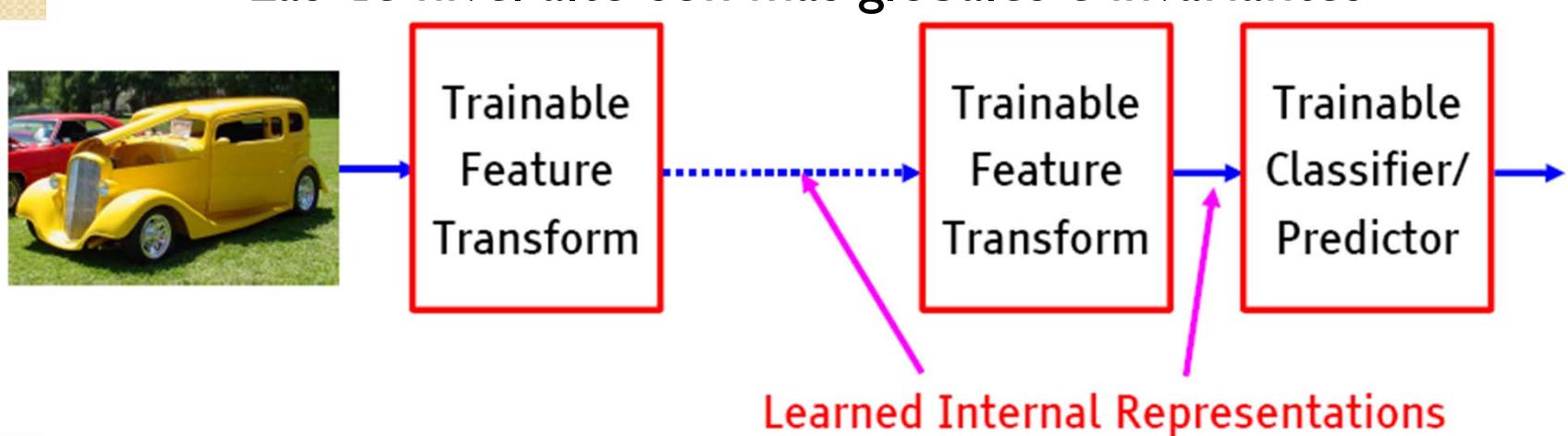
# DEEP LEARNING

- Jerarquía de representaciones con un nivel creciente de abstracción
  - Cada etapa es un tipo **entrenable** de transformación en característica
  - Ejemplos:
    - Reconocimiento de imágenes:
      - Píxel → borde → forma → parte → objeto
    - Texto
      - Carácter → palabra → grupo de palabras → sentencia → significado
    - Voz
      - Muestra → bandapectral → sonido → ... → fonema → palabra



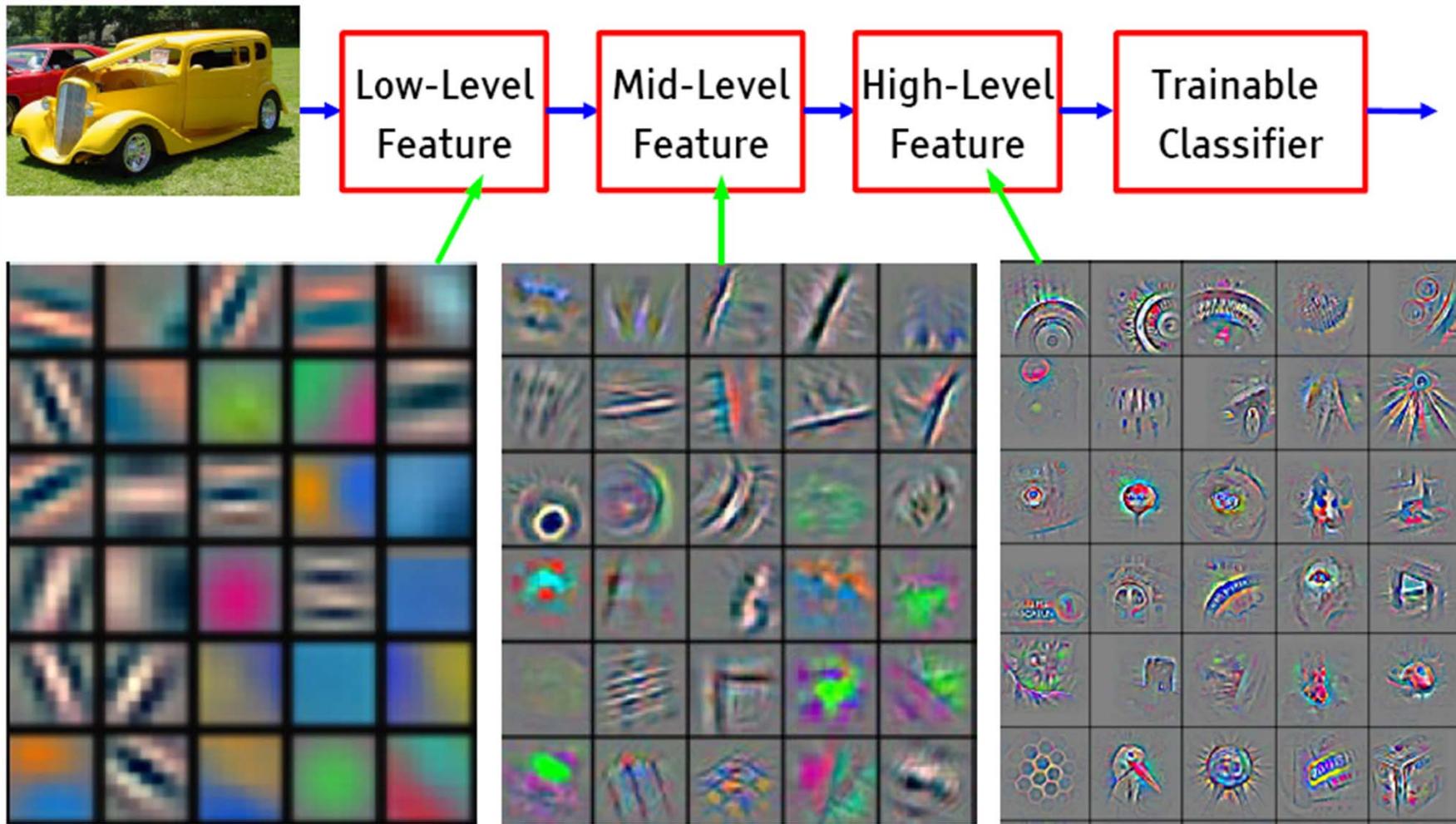
# DEEP LEARNING

- Jerarquía de transformaciones entrenables de características
  - Cada módulo transforma su representación de entrada en otra de más alto nivel
    - Las características de nivel bajo se comparten entre categorías
    - Las de nivel alto son más globales e invariantes



# DEEP LEARNING

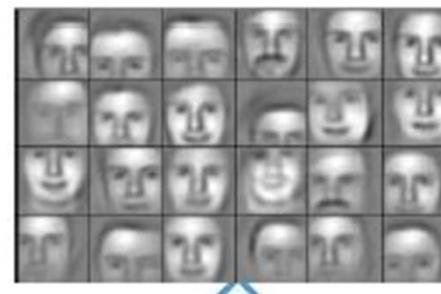
- Ejemplo:



# DEEP LEARNING

- Ejemplo:

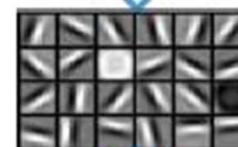
Feature representation



3rd layer  
“Objects”



2nd layer  
“Object parts”



1st layer  
“Edges”

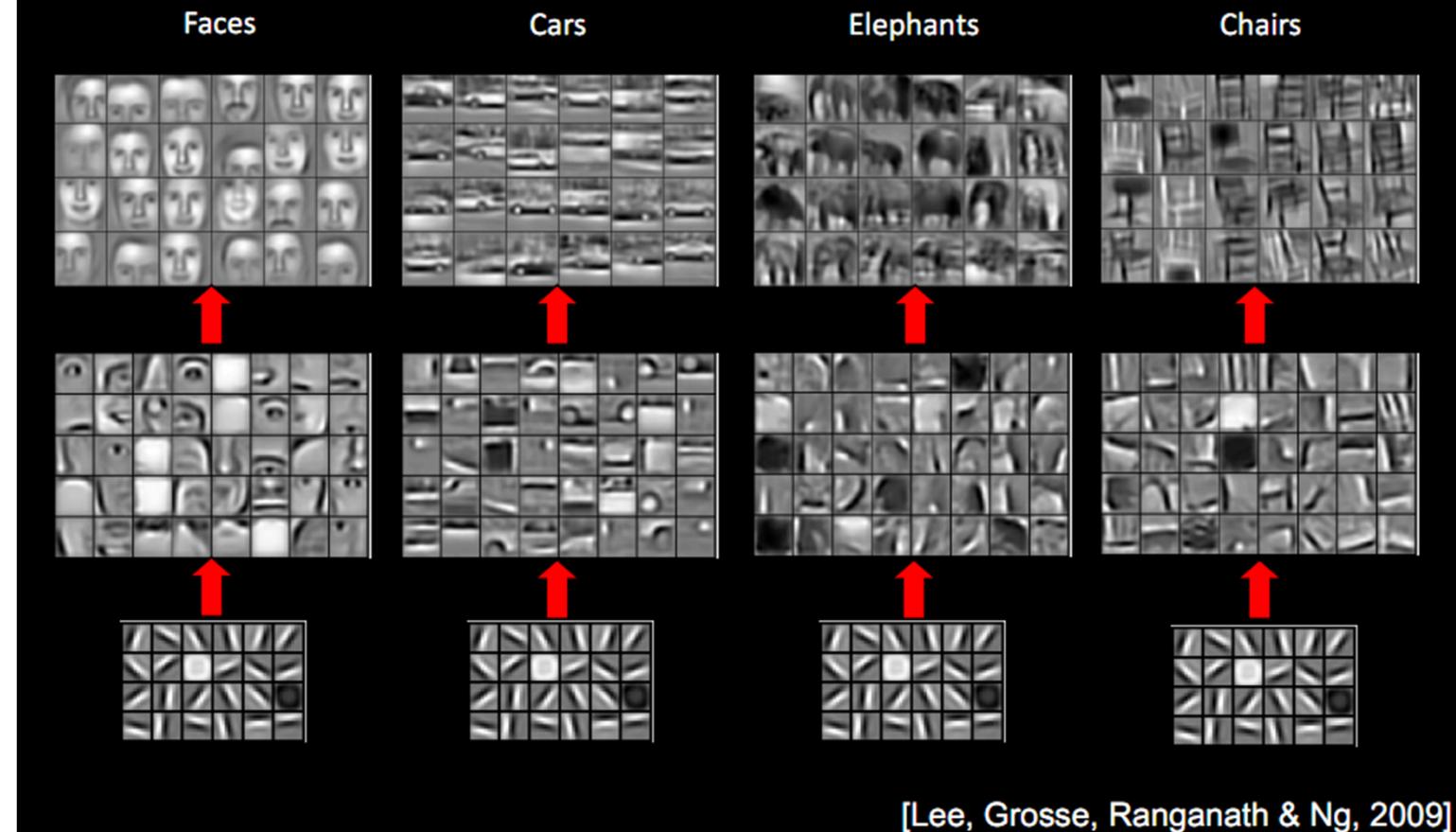


Pixels

# DEEP LEARNING

- Ejemplo:

Features learned from different object classes.





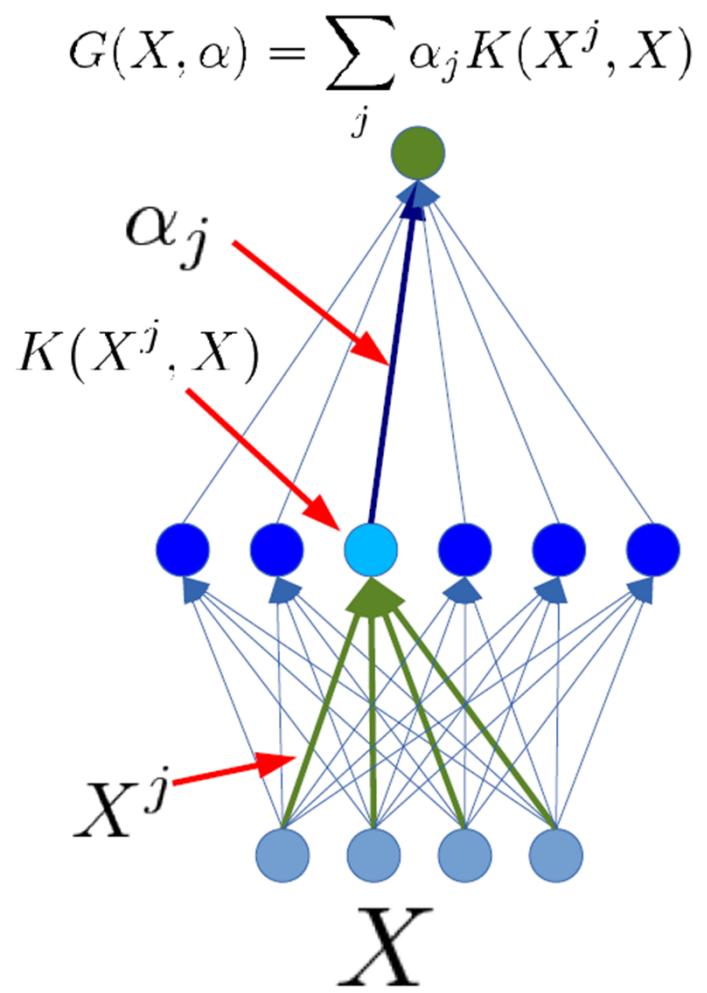
# DEEP LEARNING

- ¿Cómo se pueden desarrollar todos los módulos entrenables y que estos aprendan las representaciones apropiadas?
  - Sistema de redes de neuronas artificiales
    - Multicapa
    - Cada capa representa un módulo distinto

# DEEP LEARNING

- ¿Qué modelos son profundos?

- Las RR.NN.AA. con una capa oculta no son profundos
- Las SVM y los métodos basados en kernels no son profundos
  - Primera capa: aplicar el kernel
  - Segunda capa: lineal





# DEEP LEARNING

- ¿Qué modelos son profundos?
  - Modelos de 1-2 capas no son profundos
    - No hay jerarquía de características
  - Los árboles de clasificación no son profundos
    - No hay jerarquía de características
    - Todas las decisiones se producen en el espacio de entrada



# DEEP LEARNING

- ¿Realmente se necesitan tantas capas?
  - Teorema: cualquier función se puede aproximar con un perceptrón con una arquitectura “poco profunda”
    - Por ejemplo, perceptrones con 2 capas ocultas o máquinas basadas en kernels
      - Son “universales”
$$y = F(W^1 \cdot F(W^0 \cdot X))$$
$$y = \sum_{i=1}^P \alpha_i K(X, X^i)$$
    - Una arquitectura “profunda” sería:

$$y = F(W^K \cdot F(W^{K-1} \cdot F(\dots F(W^0 \cdot X) \dots)))$$



# DEEP LEARNING

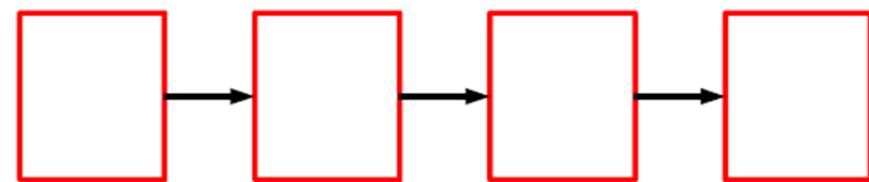
- ¿Realmente se necesitan tantas capas?
  - Las arquitecturas profundas son más eficientes para representar ciertas clases de funciones
    - Particularmente las relativas a reconocimiento visual
      - Pueden representar funciones más complejas con menos “hardware”
      - Las arquitecturas profundas cambian “espacio por tiempo” (o anchura por profundidad)
        - Más capas (más computación secuencial)
        - Pero menos hardware (menos computación paralela)

# DEEP LEARNING

- 3 tipos de arquitecturas profundas (*deep*):

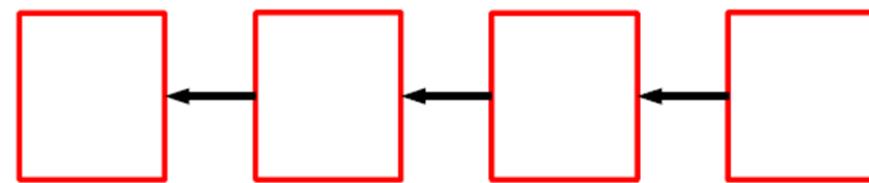
- *Feed-Forward:*

- Redes neuronales multicapa, redes convolucionales



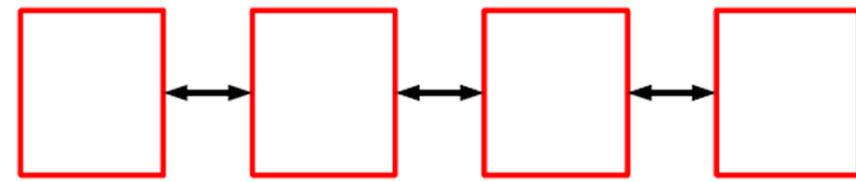
- *Feed-Back:*

- *Stacked Sparse Coding, Deconvolutional Nets*



- *Bidireccionales:*

- *Deep Boltzman Machines, Stacked Auto-Encoders*





# DEEP LEARNING

- 3 modos de entrenamiento:
  - Puramente supervisados
  - No supervisado en cada capa y supervisado en el nivel más alto
  - No supervisado en cada capa y supervisado global



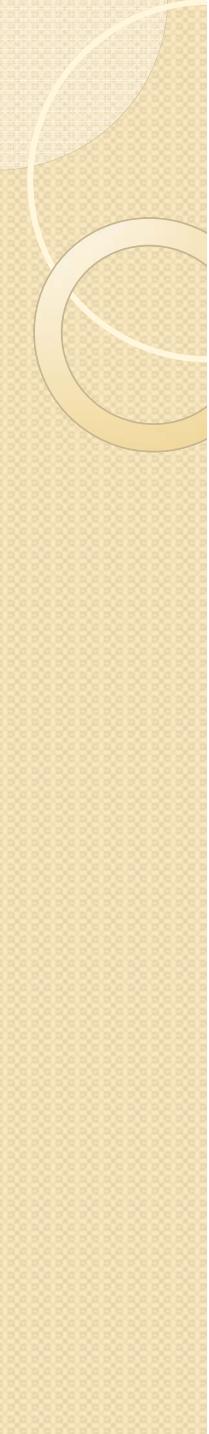
# DEEP LEARNING

- 3 modos de entrenamiento:
  - Puramente supervisados:
    - Inicializar parámetros de forma aleatoria
    - Utilizar algoritmo backpropagation para calcular el gradiente
    - Utilizado en la mayoría de sistemas de reconocimiento de voz e imágenes
  - No supervisado en cada capa y supervisado en el nivel más alto
  - No supervisado en cada capa y supervisado global



# DEEP LEARNING

- 3 modos de entrenamiento:
  - Puramente supervisados
  - No supervisado en cada capa y supervisado en el nivel más alto:
    - Entrenar cada capa de manera no supervisada, una después de la otra
    - Entrenar un clasificador supervisado en el último nivel, manteniendo las otras capas fijas
    - Funciona bien cuando hay pocas muestras etiquetadas
  - No supervisado en cada capa y supervisado global

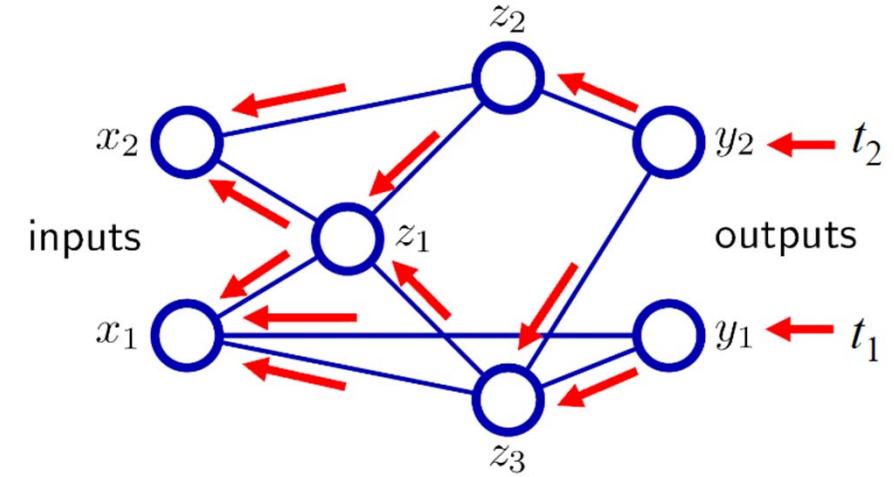
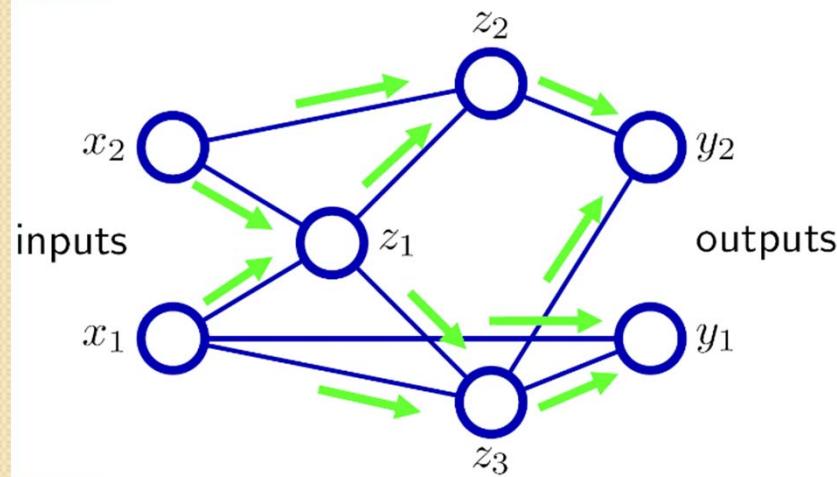


# DEEP LEARNING

- 3 modos de entrenamiento:
  - Puramente supervisados
  - No supervisado en cada capa y supervisado en el nivel más alto
  - No supervisado en cada capa y supervisado global:
    - Entrenar cada capa de manera no supervisada, una después de la otra
    - Añadir una capa supervisada, y entrenar el sistema global de forma supervisada
    - Funciona bien cuando el conjunto de etiquetas es pobre (por ejemplo, detección de peatones)

# DEEP LEARNING

- Entrenamiento de una red neuronal:
  - El algoritmo de *backpropagation* puede utilizarse para entrenar una red neuronal con un número arbitrario de capas ocultas
    - Incluso arquitecturas no divididas en capas “convencionales”



# DEEP LEARNING

- Entrenamiento de una red neuronal:
  - Sin embargo, el *backpropagation* no funciona bien si los pesos se inicializan de forma aleatoria:
    - Algunos estudios indican que las redes profundas entrenadas con *backpropagation* dan peores resultados que redes “poco profundas”

	Experiment 2			Experiment 3		
	train.	valid.	test	train.	valid.	test
DBN, unsupervised pre-training	0%	1.2%	1.2%	0%	1.5%	1.5%
Deep net, auto-associator pre-training	0%	1.4%	1.4%	0%	1.4%	1.6%
Deep net, supervised pre-training	0%	1.7%	2.0%	0%	1.8%	1.9%
Deep net, no pre-training	.004%	2.1%	2.4%	.59%	2.1%	2.2%
Shallow net, no pre-training	.004%	1.8%	1.9%	3.6%	4.7%	5.0%

- Bengio et al., NIPS 2007



# DEEP LEARNING

- Entrenamiento de una red neuronal:
  - Problemas del backpropagation:
    - El gradiente se va diluyendo progresivamente
      - Más allá de las capas superiores, la señal corrección es mínima
      - Se queda parado en mínimos locales
        - Especialmente si se comienza la búsqueda alejado de “buenas” regiones (inicialización aleatoria)
        - Conveniente una etapa anterior de **preentrenamiento** para fijar un buen punto de partida
          - Valores iniciales de los pesos
      - Generalmente, sólo se puede usar datos etiquetados
        - Casi todos los datos son no etiquetados
        - El cerebro puede aprender de datos no etiquetados
        - Sería conveniente utilizar un sistema de aprendizaje no supervisado o semisupervisado para aprovechar todos estos datos

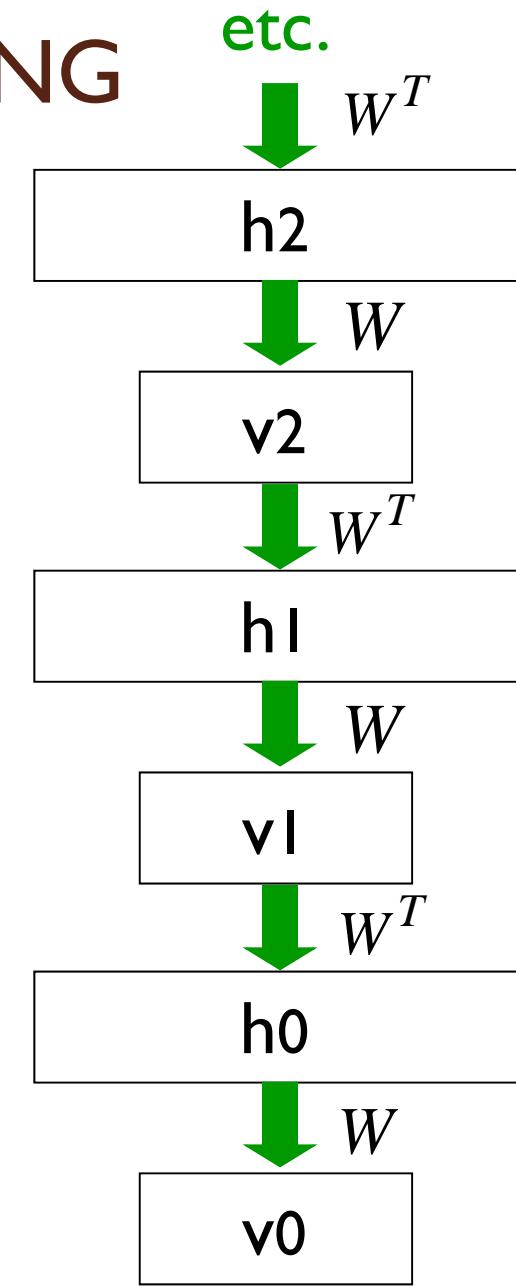
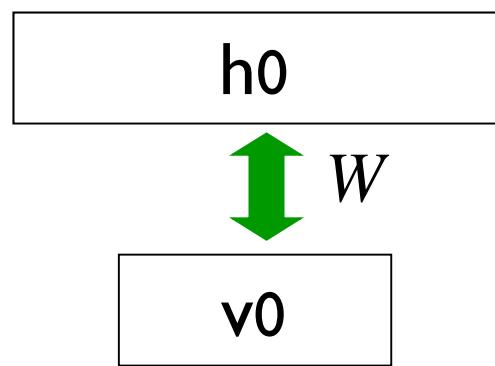


# DEEP LEARNING

- Entrenamiento de una red neuronal:
  - Una forma de resolver estos problemas es utilizar el *greedy layer-wise training*:
    - I - Utiliza aprendizaje no supervisado capa por capa
      - Permite que se desarrolle la abstracción de forma natural de una capa a otra
      - Proceso:
        - Entrenar la primera capa de forma no supervisada utilizando los datos sin las etiquetas
        - “Congelar” esos pesos recién entrenados, y entrenar la segunda capa de forma no supervisada
          - Utiliza la salida de la primera cada como entrada a la capa no supervisada
          - Repetir este proceso para tantas capas como se quiera
        - Provee de un conjunto de pesos a cada capa
        - En el proceso de entrenamiento de cada capa, se puede usar también abundantes datos no etiquetados que no sean parte del conjunto de entrenamiento

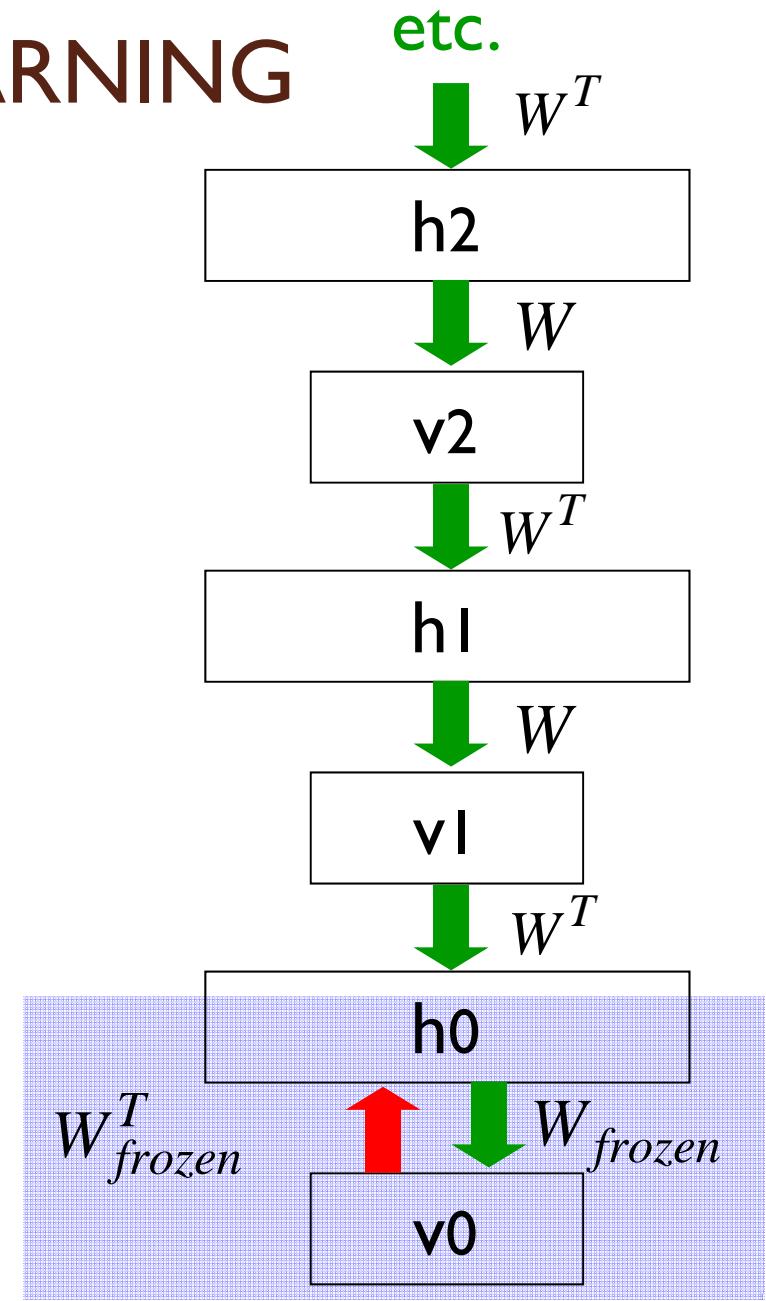
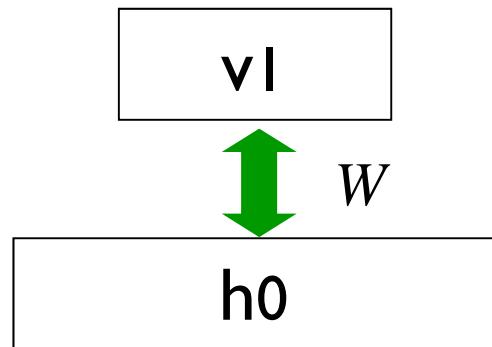
# DEEP LEARNING

- Entrenar:



# DEEP LEARNING

- Entrenar:





# DEEP LEARNING

- Entrenamiento de una red neuronal:
  - Una forma de resolver estos problemas es utilizar el *greedy layer-wise training*:
    - 2 - Usar las salidas de la capa final como entradas de una capa/modelo supervisado y entrenar la última capa de forma supervisada
      - Dejando los pesos de las capas anteriores “congelados”
    - 3 - Realiza un entrenamiento supervisado de arriba a abajo como etapa final
      - Descongela y modifica todos los pesos
      - Los valores iniciales de los pesos son los resultantes de las etapas anteriores
        - No son aleatorios
        - Buena zona para comenzar la búsqueda
      - Refina las capas intermedias para que sean más relevantes para el trabajo a realizar



# DEEP LEARNING

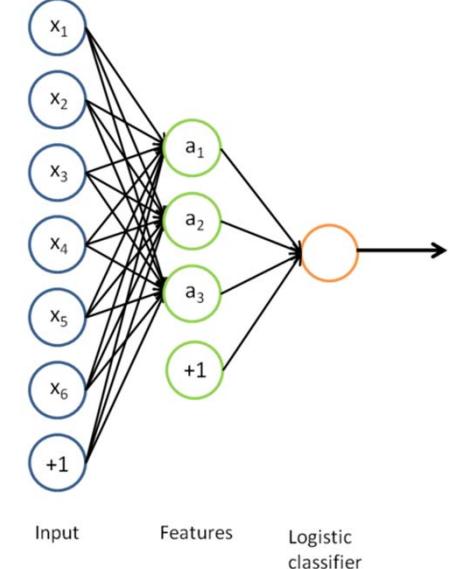
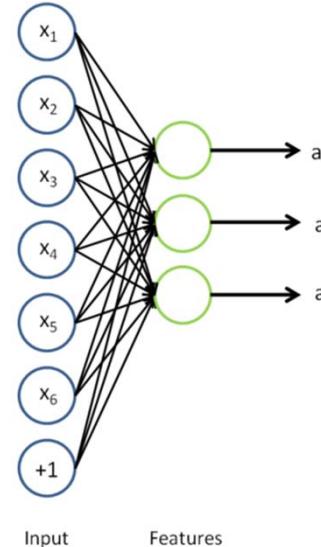
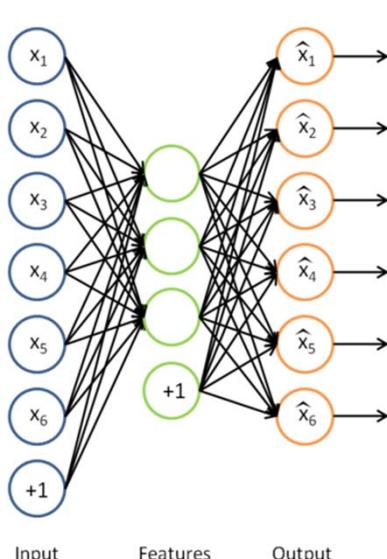
- Entrenamiento de una red neuronal:
  - Una forma de resolver estos problemas es utilizar el *greedy layer-wise training*:
    - Cada capa recibe toda la atención del proceso de entrenamiento de cada vez, por turnos
    - Se puede utilizar los datos no etiquetados
    - Cuando finalmente se entrena la red entera con entrenamiento supervisado, los pesos ya han sido ajustados, con lo que ya se está en un buen nivel de error y sólo se necesita un ajuste fino
      - Esto ayuda con los problemas de
        - Aprendizaje de cada capa ineficaz
        - Mínimos locales
      - Dos aproximaciones comunes:
        - *Stacked Auto-Encoders*
        - *Deep Belief Networks*

# DEEP LEARNING

- *Stacked auto-encoders:*

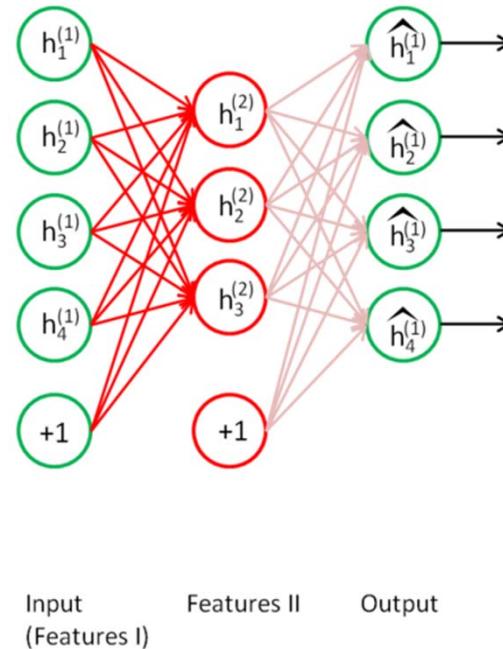
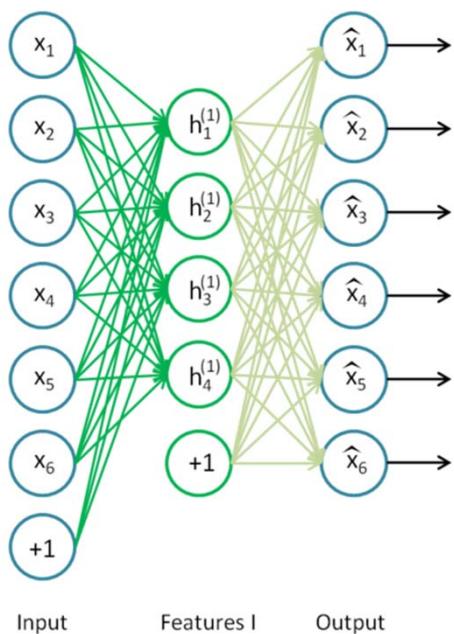
- *Auto-Encoders:*

- Tipo de aprendizaje no supervisado que intenta descubrir características genéricas en los datos
    - Aprender la función identidad mediante el aprendizaje de características importantes
      - No sólo pasando los datos
    - Compresión de la información
    - Puede usar las nuevas características en el conjunto de entrenamiento o concatenar ambas



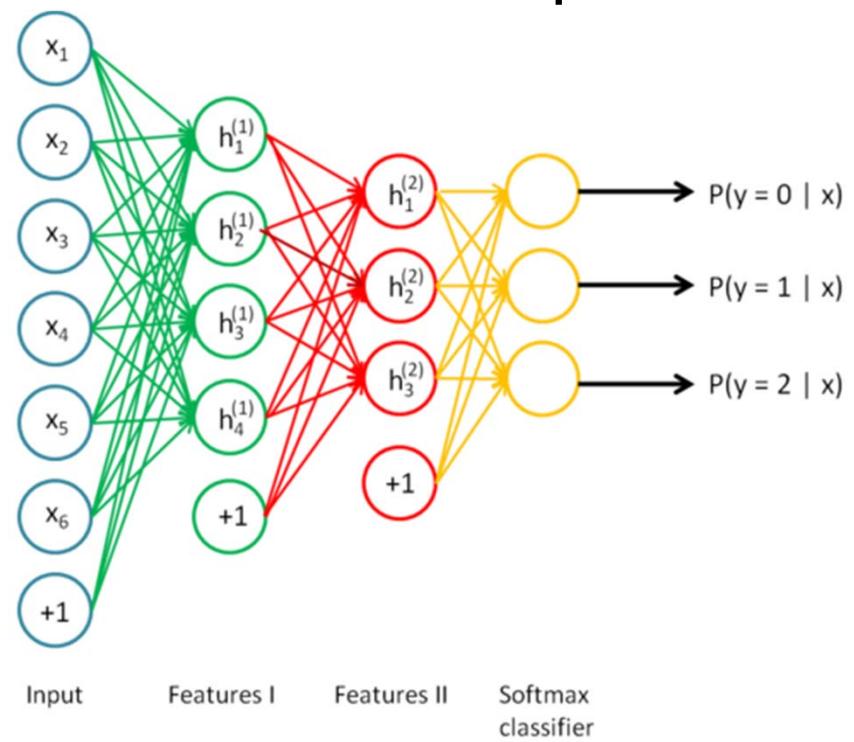
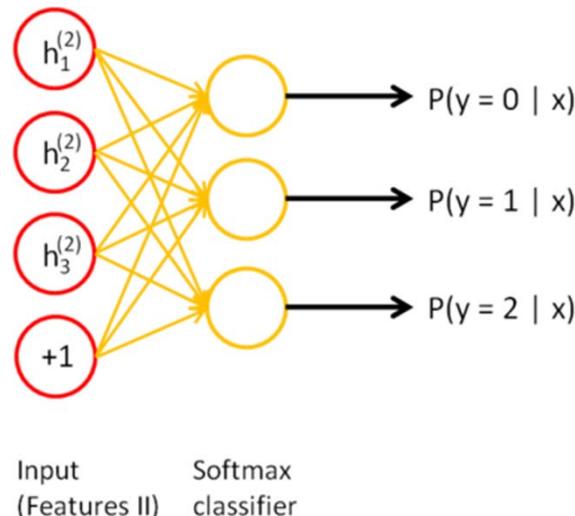
# DEEP LEARNING

- *Stacked auto-encoders:*
  - Bengio (2007)
  - Apilar muchos *auto-encoders* en serie y entrenarlos utilizando *greedy layer-wise training*
    - De cada vez que se entrena, poner la capa de “decodificación” de salida



# DEEP LEARNING

- *Stacked auto-encoders:*
  - Realizar entrenamiento supervisado en la última capa usando las características finales
  - Despues, realizar entrenamiento supervisado en toda la red para ajustar los valores de los pesos





# DEEP LEARNING

- *Stacked auto-encoders:*
  - Realizan una reducción de la dimensionalidad
    - Reducción que puede ser similar a PCA o no lineal
  - Esto lleva a una representación “densa” en un conjunto de características
    - Para una entrada, generalmente todas sus características tendrán valores distintos de 0
      - Contienen la información comprimida
    - Esto hace que para las capas sucesivas sea más difícil extraer las características más importantes
  - Solución: *sparse encoders*

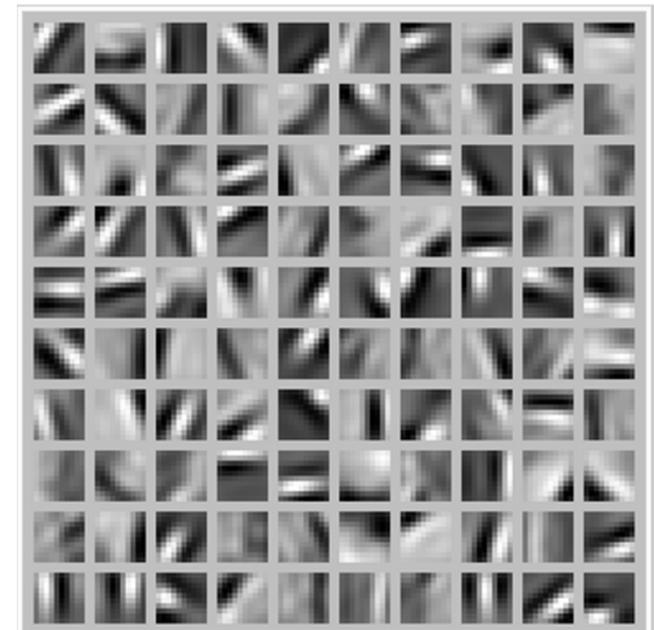


# DEEP LEARNING

- *Stacked auto-encoders:*
  - *Sparse encoders*
    - En cada capa se utilizan más características de forma que, ante una entrada, la mayoría de ellas tomen un valor de 0
    - Cada característica indicará la pertenencia o no a esa característica/clase
    - Permite un entrenamiento más rápido en capas posteriores

# DEEP LEARNING

- *Stacked auto-encoders:*
  - *Sparse encoders*
    - Ejemplo:
      - Para las siguientes, es más fácil de ver cómo se representa una imagen si sólo unas pocas de ellas tienen un valor distinto de 0 que si todas tienen un valor distinto de 0:
      - Las máquinas aprenden más fácilmente de la primera manera





# DEEP LEARNING

- *Stacked auto-encoders:*
  - *Sparse encoders*
    - Implementación:
      - Usar más nodos ocultos
      - Usar técnicas de regularización que promueven este tipo de características, por ejemplo:
        - Modificar la función de aprendizaje para penalizar nodos distintos de 0
        - Decaimiento de pesos
        - etc.



# DEEP LEARNING

- *Stacked auto-encoders:*
  - *Denoising Auto-Encoders*
    - Corromper estocásticamente la instancia de entrenamiento de cada vez, pero entrenar el *auto-encoder* para decodificar la instancia corrupta, forzándolo a aprender dependencias condicionales dentro de la instancia
    - Buenos resultados empíricos
    - Maneja bien si faltan datos (*missing values*)

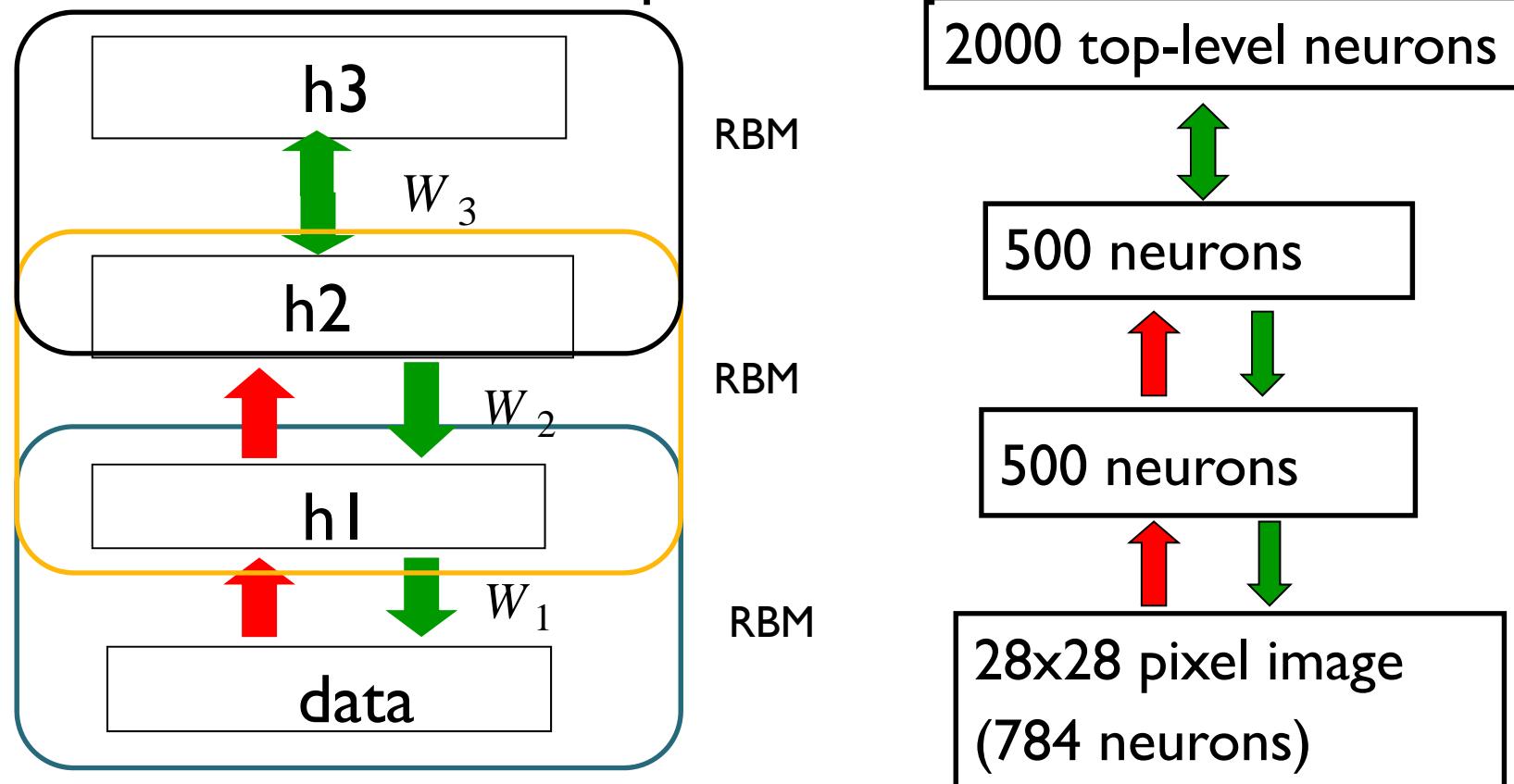


# DEEP LEARNING

- *Stacked auto-encoders:*
  - Etapa final de entrenamiento supervisado de toda la red:
    - Realizarla si se tiene una cantidad suficiente de datos etiquetados
    - Si no se realiza, una opción podría ser concatenar las entradas con las características de las capas ocultas en la capa de salida
  - *Stacked auto-encoders* empíricamente no ofrecen tan buenos resultados como *Deep Belief Networks* (DBNs)

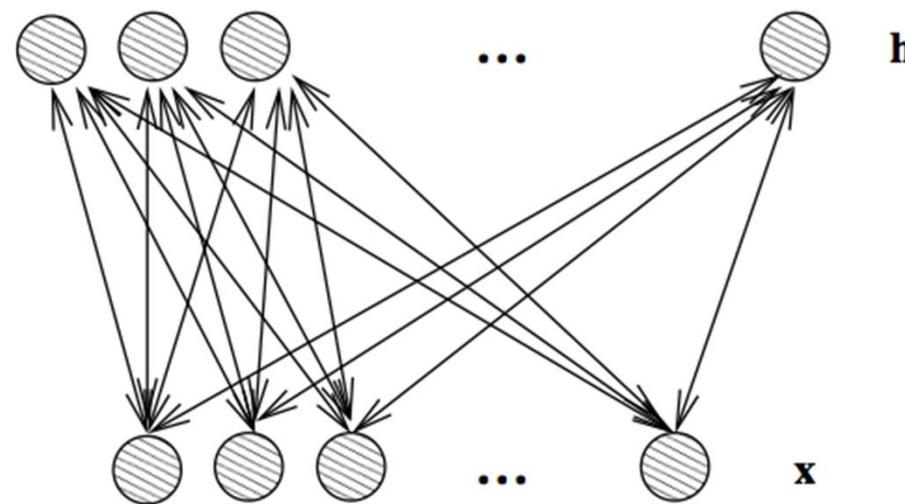
# DEEP LEARNING

- Deep Belief Networks (DBN)
  - Son pilas de RBMs (Restricted Boltzman machine) formando una arquitectura profunda



# DEEP LEARNING

- Deep Belief Networks (DBN)
  - Geoff Hinton (2006)
  - Utiliza entrenamiento por capa (*greedy layer-wise training*) pero cada capa es una RBM (*Restricted Boltzmann Machine*)
    - *Boltzmann machine* con restricciones:





# DEEP LEARNING

- *Deep Belief Networks (DBN)*
  - Geoff Hinton (2006)
  - Utiliza entrenamiento por capa (*greedy layer-wise training*) pero cada capa es una RBM (*Restricted Boltzmann Machine*)
    - *Boltzmann machine* con restricciones:
      - No hay conexiones laterales entre los nodos ocultos (h) y los visibles (x)
      - Pesos simétricos
      - No utiliza alineamiento/temperatura
      - Típicamente, utiliza nodos logísticos probabilísticos, pero otras funciones de activación son posibles



# DEEP LEARNING

- *Deep Belief Networks (DBN)*
  - Una RBM es un modelo basado en energía
    - Los modelos basados en energía definen la distribución de probabilidad a través de una función de energía:

$$P(v, h) = \frac{e^{-Energy(v, h)}}{\sum_x e^{-Energy(v, h)}}$$

$$Energy(v, h) = \sum_i f_i(v, h)$$

“f” es el experto

# DEEP LEARNING

- Deep Belief Networks (DBN)
  - Una Boltzman machine conecta neuronas binarias estocásticas usando conexiones simétricas

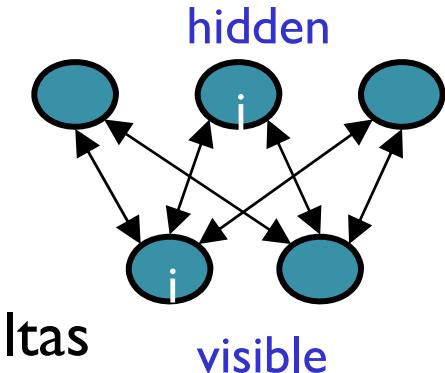
$$Energy(v, h) = -b'v - c'h - h'Wv - v'Uv - h'Vh$$

b, c son bias de x, h, W, U, V son pesos

$$p(v, h) \propto e^{-Energy(v, h)}$$

- Una RBM restringe la conectividad para hacer el aprendizaje más fácil
  - Sólo una capa oculta
  - No hay conexiones entre unidades ocultas

$$Energy(v, h) = -b'v - c'h - h'Wv$$



# DEEP LEARNING

- Deep Belief Networks (DBN)
  - Cómputo de la energía en una RBM:

Estado binario de la unidad visible i      Estado binario de la unidad oculta j

$$E(v, h) = - \sum_{i,j} v_i h_j w_{ij}$$

Energía con la configuración v en las unidades visibles y h en las ocultas

peso entre las unidades i y j

- Aproximación del gradiente:
  - Contrastive divergence:

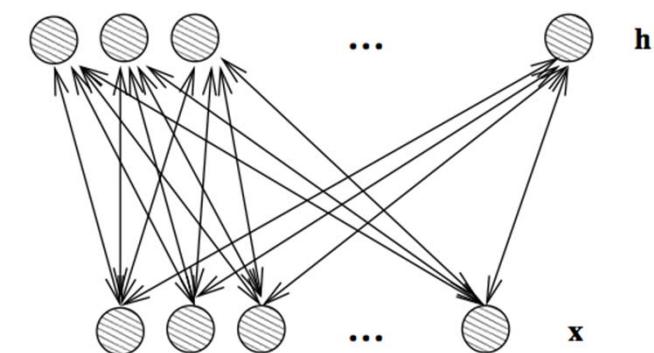
$$\frac{\partial E(v, h)}{\partial w_{ij}} = - v_i h_j$$

# DEEP LEARNING

- *Deep Belief Networks (DBN)*

- Entrenamiento de cada RBM:

- Estado inicial se asigna a un ejemplo de entrenamiento  $x$  (puede tomar valores reales)
    - Se realiza un proceso iterativo hacia delante y atrás
      - $P(h_i = 1|x) = \text{sigmoid}(W_i x + c_i) = 1/(1+e^{-\text{net}(hi)})$
      - $c_i$  es el bias de un nodo oculto
      - $P(x_i = 1|h) = \text{sigmoid}(W'_i h + b_i) = 1/(1+e^{-\text{net}(xi)})$
      - $b_i$  es el bias de un nodo visible



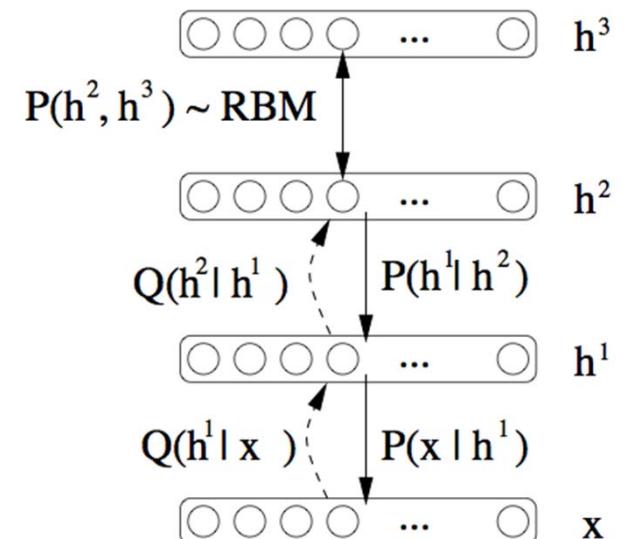


# DEEP LEARNING

- *Deep Belief Networks (DBN)*
  - Entrenamiento de cada RBM:
    - Estado inicial se asigna a un ejemplo de entrenamiento  $x$  (puede tomar valores reales)
    - Se realiza un proceso iterativo hacia delante y atrás
      - Se calcula así la diferencia entre el valor de  $x$  y el valor original del patrón de entrenamiento  $x$  tras  $k$  iteraciones: CD- $k$  (*Contrastive Divergence*)
    - Actualizar los pesos para decrementar la divergencia como en una máquina de Boltzman
    - Típicamente, sólo hace CD-1
      - Buenos resultados empíricos

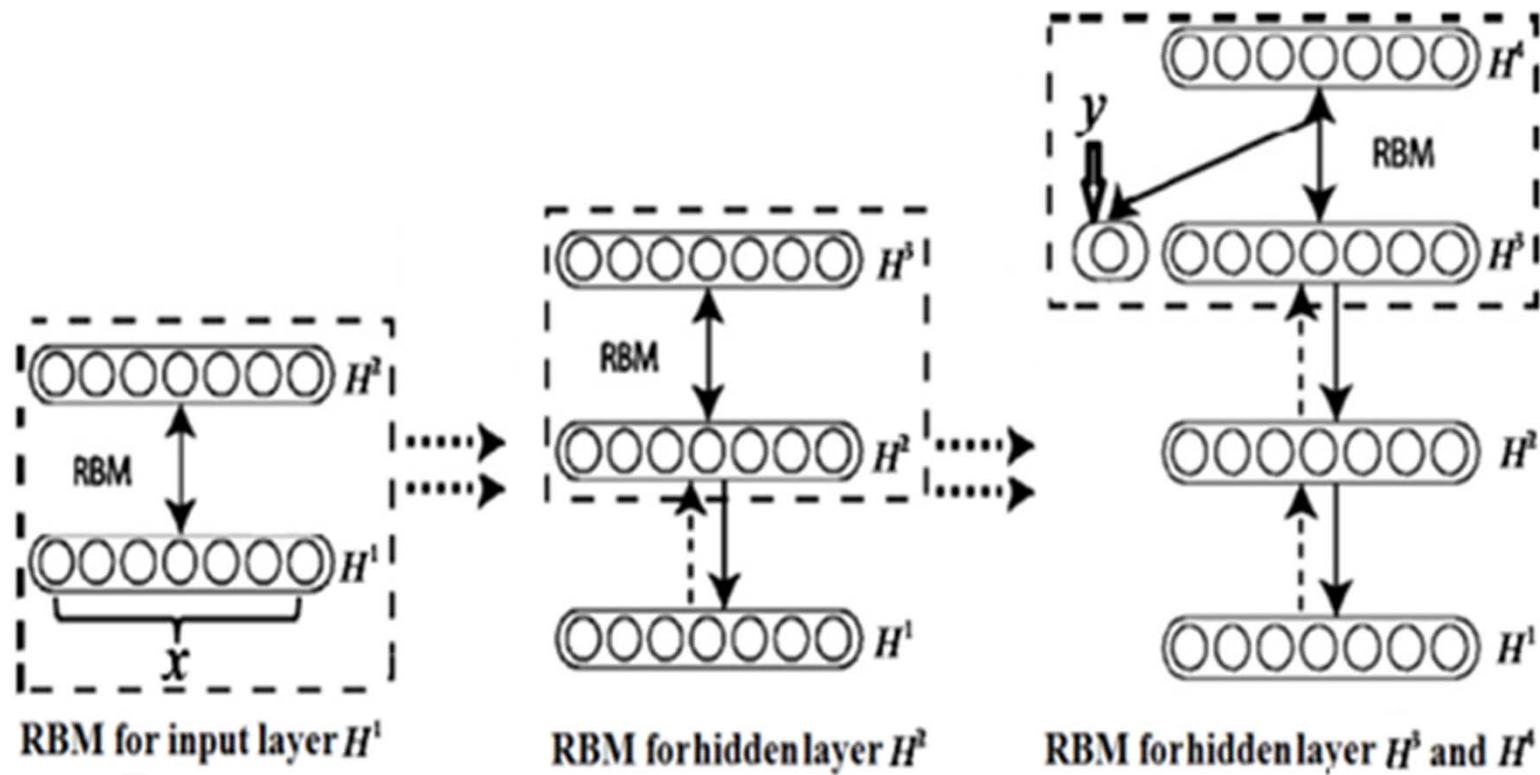
# DEEP LEARNING

- *Deep Belief Networks (DBN)*
  - Entrenamiento de la DBN:
    - Misma aproximación capa por capa:
      - Primero entrenar la RBM más baja ( $h^0 - h^1$ )
        - $h^0$  es  $x$
      - Congelar los pesos y entrenar las capas RBM siguientes
      - Después, conectar las salidas finales a un modelo supervisado que entrene todo ese modelo (la última capa)
      - Finalmente, descongelar todos los pesos, y entrenar el global DBN con un modelo supervisado que ajuste en fino todos los pesos



# DEEP LEARNING

- Deep Belief Networks (DBN)
- Entrenamiento de la DBN:
  - Misma aproximación capa por capa:



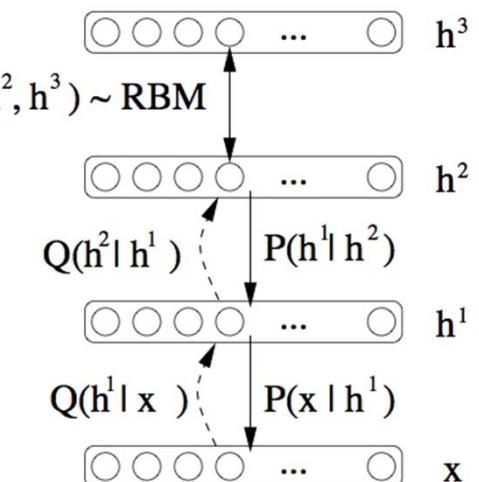


# DEEP LEARNING

- *Deep Belief Networks (DBN)*
  - Esta aproximación no supervisada aprende características de los datos (en las capas ocultas de los RBMs) que pueden ayudar en el proceso de hacer más adecuados los tipos de los patrones del conjunto de entrenamiento.
  - Esto descubre características que pueden estar asociadas en varios patrones de entrenamiento, y por tanto pueden ser potencialmente útiles para otros objetivos en el conjunto de entrenamiento
    - Clasificación
    - Compresión
    - etc.

# DEEP LEARNING

- *Deep Belief Networks (DBN)*
  - Puede ser utilizada para crear vectores de ejemplo  $x$ 
    - Inicializar la capa superior a un valor arbitrario
      - Iterar entre las dos capas superiores  $m$  veces
    - Pasar el vector hacia abajo en la red
    - El último vector en la capa inferior es el  $x$  generado
      - Puede ser de valores reales
    - De forma alternativa, se podría empezar con un vector  $x$ , pasarlo a la capa de arriba, y traerlo de nuevo a la de abajo para generar un nuevo  $x$





# DEEP LEARNING

- *Deep Belief Networks (DBN)*

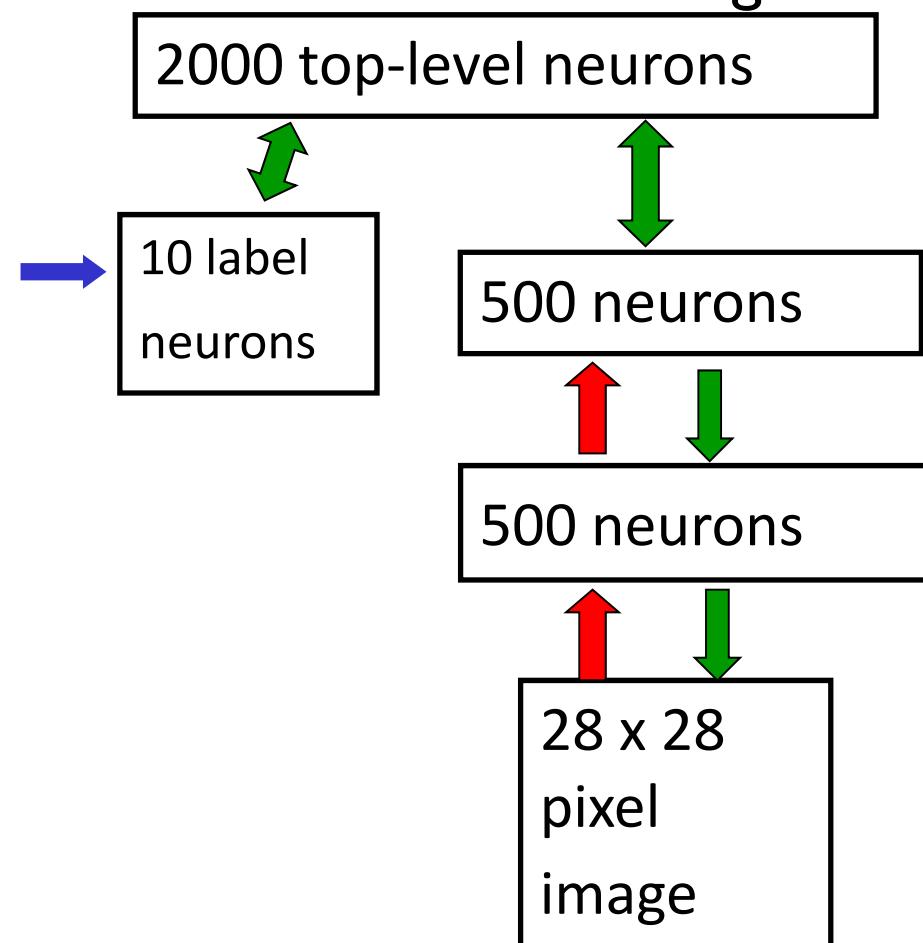
- Además:

- Se pueden usar conexiones laterales en los RBMs (dejarían de ser RBMs)
      - Pueden dar mejores resultados
    - *Deep Boltzman machine*
      - Permite relajación continua a través de toda la red:
        - Recibir entradas por las capas superior e inferior en lugar de una secuencia a través de capas RBM
        - Típicamente, para entrenar con éxito, primero inicializar los pesos utilizando el entrenamiento DBN con capas RBM
      - *Contitional RBM / Temporal RBM*
        - Permiten que las probabilidades de los nodos puedan ser condicionadas por otras entradas: contexto, recurrencia, etc.

# DEEP LEARNING

- *Deep Belief Networks (DBN)*
  - Un modelo neuronal de reconocimiento de dígitos:

Cuando se entrena la capa superior de pesos, las etiquetas fueron dadas como parte de la entrada



# DEEP LEARNING

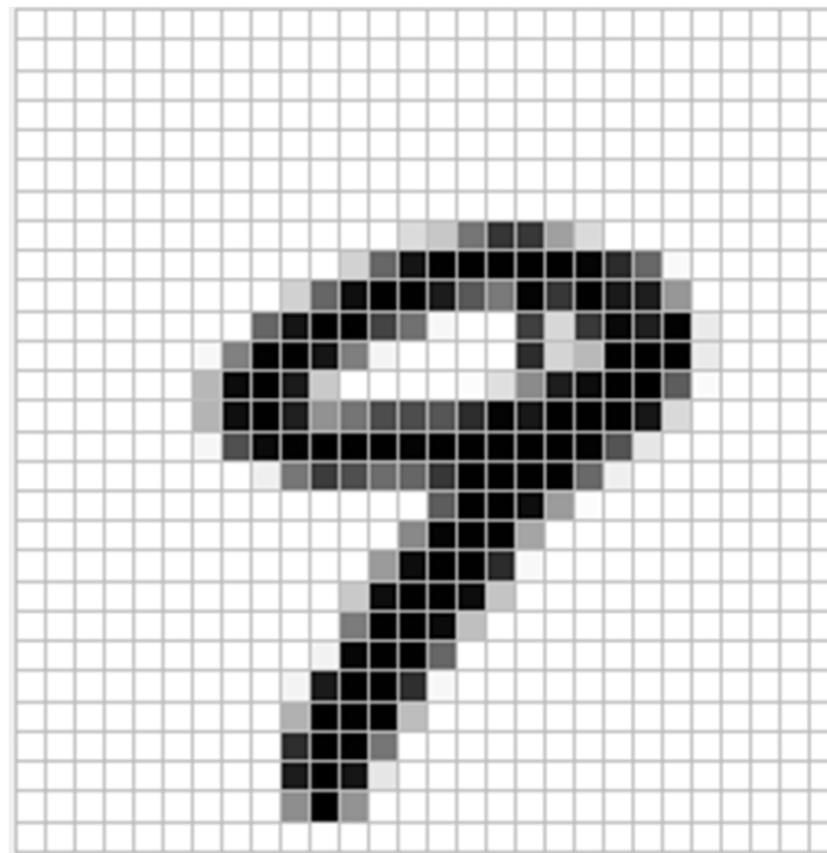
- *Deep Belief Networks (DBN)*
  - Un modelo neuronal de reconocimiento de dígitos:
    - Resultados:

<b>Generative model based on RBM's</b>	<b>1.25%</b>
Support Vector Machine (Decoste et. al.)	1.4%
Backprop with 1000 hiddens (Platt)	~1.6%
Backprop with 500 -->300 hiddens	~1.6%
K-Nearest Neighbor	~ 3.3%

- Número de imágenes de entrenamiento: 60,000
- Número de imágenes de test: 10,000
- Tiempo total de entrenamiento: una semana

# DEEP LEARNING

- *Deep Belief Networks (DBN)*
  - Un modelo neuronal de reconocimiento de dígitos:



# DEEP LEARNING

- *Deep Belief Networks (DBN)*
    - Un modelo neuronal de reconocimiento de dígitos:
      - Ejemplos generados dejando a la memoria asociativa ejecutarse con una etiqueta fijada

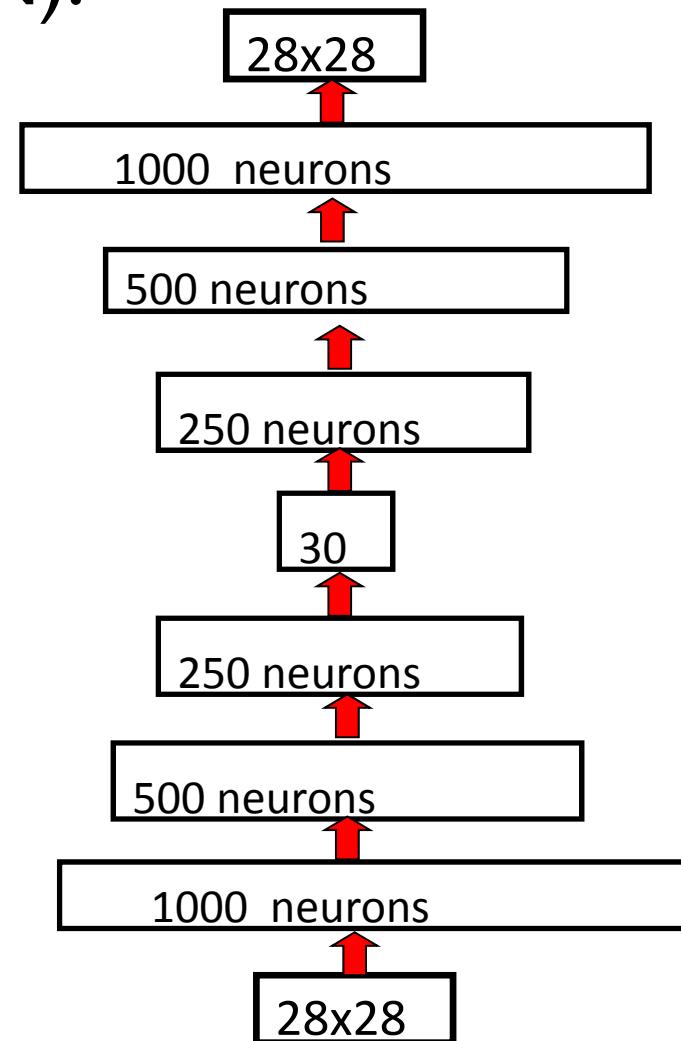
# DEEP LEARNING

- *Deep Belief Networks (DBN)*
  - Un modelo neuronal de reconocimiento de dígitos:
    - Dando una imagen binaria aleatoria como entrada:

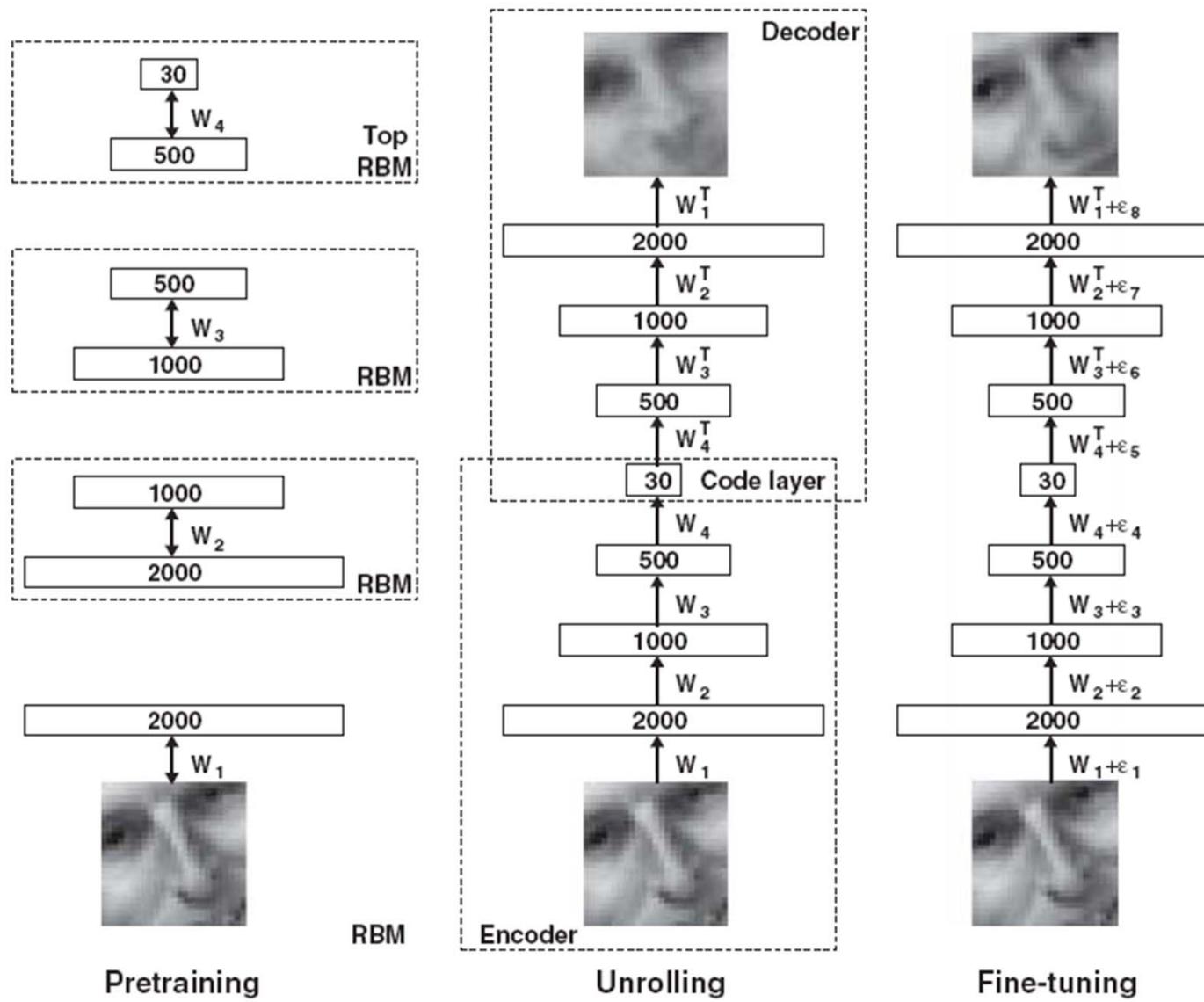


# DEEP LEARNING

- *Deep Belief Networks (DBN):*
  - Reducir la dimensionalidad de los datos:
    - Entrenar una pila de 4 RBMs
    - “Desenrollarlos”
    - Ajustar los pesos con backpropagation



# DEEP LEARNING



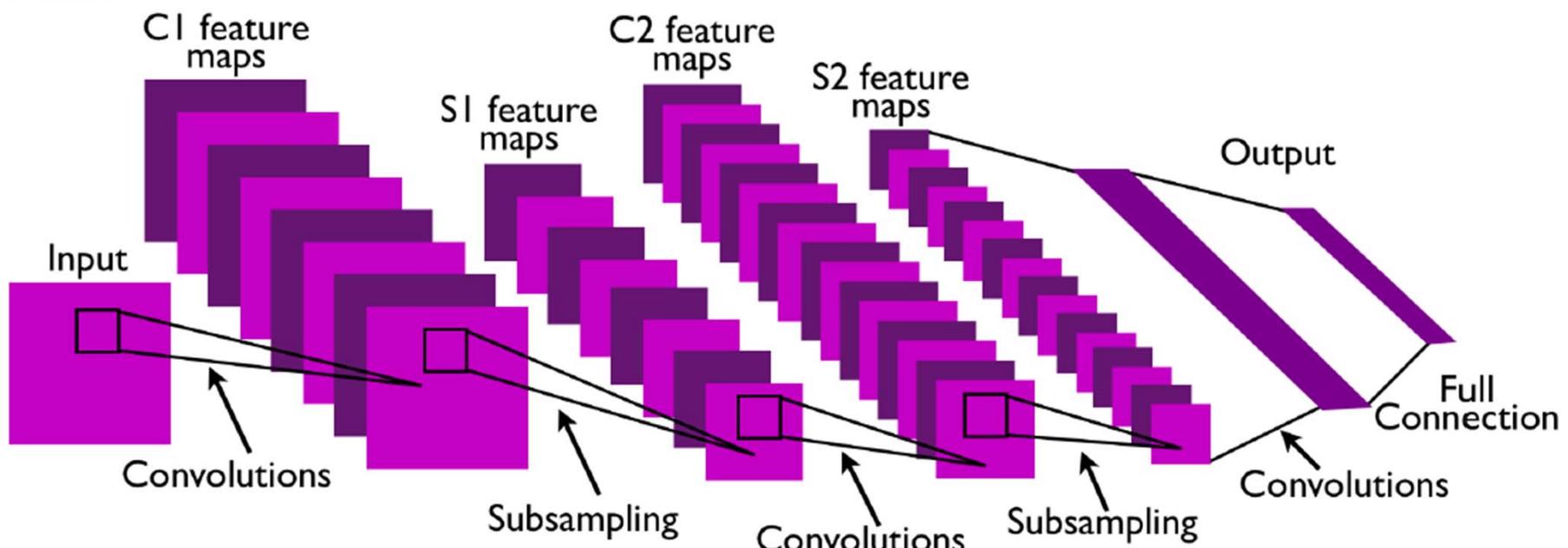


# DEEP LEARNING

- *Convolutional neural networks:*
  - Diseñadas para reconocer patrones visuales directamente de los píxeles, con un preprocessado mínimo
  - Son capaces de reconocer patrones con variabilidad extrema, y de forma robusta a las distorsiones y a transformaciones geométricas simples

# DEEP LEARNING

- *Convolutional neural networks:*
  - Estructura:





# DEEP LEARNING

- *Convolutional neural networks:*
  - Las capas pueden ser de 3 tipos distintos:
    - Convolucionales
    - *Max-Pooling*
    - Totalmente conectadas



# DEEP LEARNING

- *Convolutional neural networks:*
  - Las capas pueden ser de 3 tipos distintos:
    - Convolucionales
      - Malla de neuronas formando un rectángulo
      - La capa anterior debe de ser otra malla rectangular de neuronas
        - Imagen
      - Cada neurona toma entradas de una sección rectangular de la capa anterior
        - Los pesos de todas las neuronas son los mismos
      - Cada neurona, por tanto, realiza una operación de convolución en parte de la capa anterior
        - Los pesos especifican el filtro de convolución
      - Además, cada capa puede constar de varias mallas
        - Cada malla toma entradas de todas las mallas de la capa anterior, utilizando filtros potencialmente diferentes
    - *Max-Pooling*
    - Totalmente conectadas

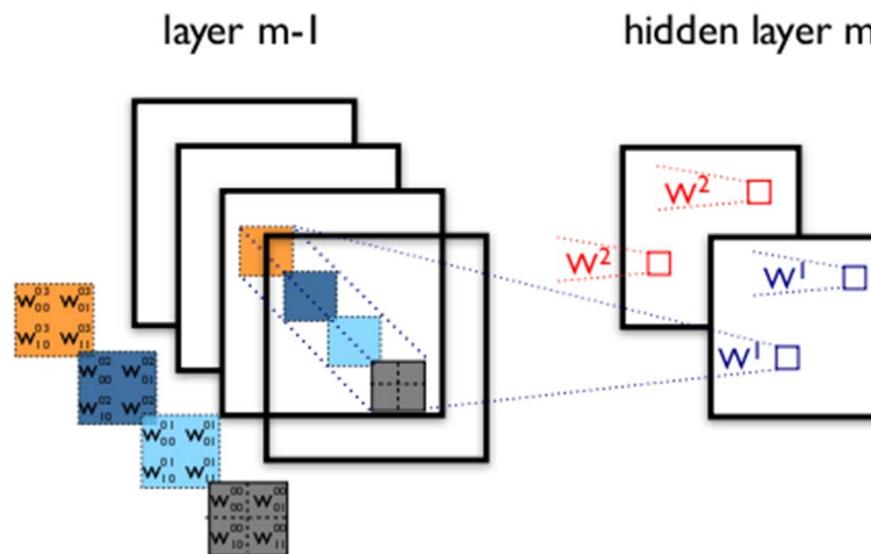


# DEEP LEARNING

- *Convolutional neural networks:*
  - Las capas pueden ser de 3 tipos distintos:
    - Convolucionales
      - Si una capa convolucional tiene como entrada una capa cuadrada de NxN neuronas, y la capa convolucional implementa un filtro mxm
      - La salida de la capa convolucional será de tamaño:
        - $(N-m) \times (N-m)$
    - *Max-Pooling*
    - Totalmente conectadas

# DEEP LEARNING

- *Convolutional neural networks:*
  - Las capas pueden ser de 3 tipos distintos:
    - Convolucionales



- *Max-Pooling*
- Totalmente conectadas



# DEEP LEARNING

- *Convolutional neural networks:*
  - Las capas pueden ser de 3 tipos distintos:
    - Convolucionales
    - *Max-Pooling*
      - Toma bloques rectangulares pequeños de la capa convolucional anterior y los submuestrea para producir una única salida de ese bloque
      - Reduce un bloque rectangular a un valor
      - Hay varias formas de hacer esto: tomar media, máximo, combinación lineal, etc.
    - Totalmente conectadas



# DEEP LEARNING

- *Convolutional neural networks:*
  - Las capas pueden ser de 3 tipos distintos:
    - Convolucionales
    - *Max-Pooling*
      - Si la capa anterior es de tamaño  $N \times N$  y cada neurona toma una región de tamaño  $k \times k$ , esta capa tendrá una salida de tamaño  $\frac{N}{k} \times \frac{N}{k}$
    - Totalmente conectadas

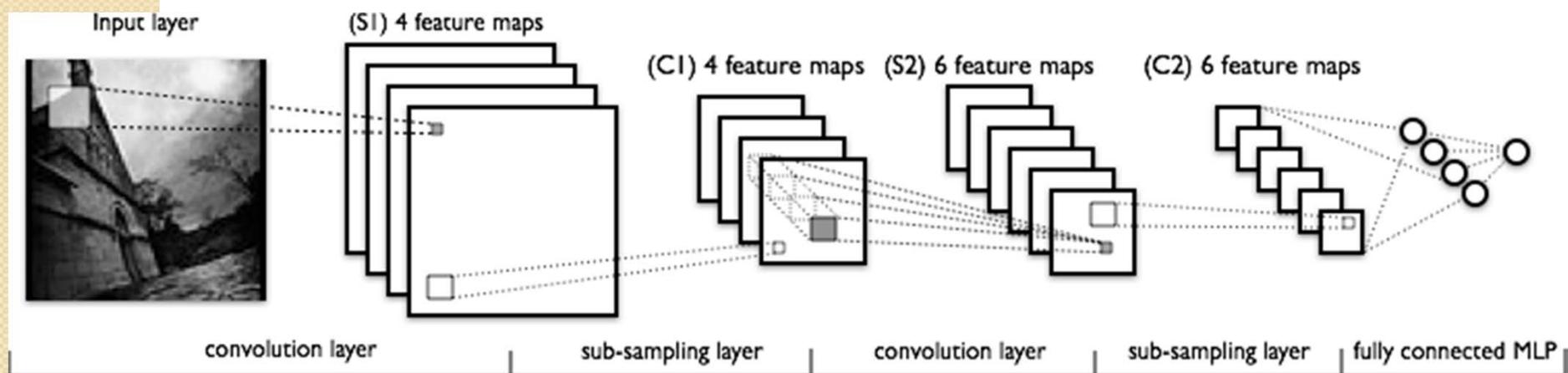


# DEEP LEARNING

- *Convolutional neural networks:*
  - Las capas pueden ser de 3 tipos distintos:
    - Convolucionales
    - *Max-Pooling*
    - Totalmente conectadas
      - Capas finales de la red, con el razonamiento de más alto nivel
        - MLP (perceptrón multicapa)
      - Toma todas las neuronas de la capa anterior y las conecta a cada neurona de salida que tenga
      - Ya no son capas localizadas espacialmente

# DEEP LEARNING

- *Convolutional neural networks:*
  - Estructura (ejemplo):



- Capa convolucional: extraer mapas de características a partir de una imagen anterior
  - Cada malla, un mapa de una característica distinta
  - En base a convoluciones



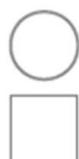
# DEEP LEARNING

- *Convolutional neural networks:*
  - Además:
    - No está limitado a ser usado con entradas de 2 dimensiones (imágenes)
    - Se pueden construir también redes convolucionales de 1, 3, etc. dimensiones
    - Ejemplo:
      - Para audio (1 dimensión)
      - Para MRI (3 dimensiones)
    - Construir filtros y capas de submuestreo de esa dimensión

# DEEP LEARNING

- *Convolutional neural networks:*

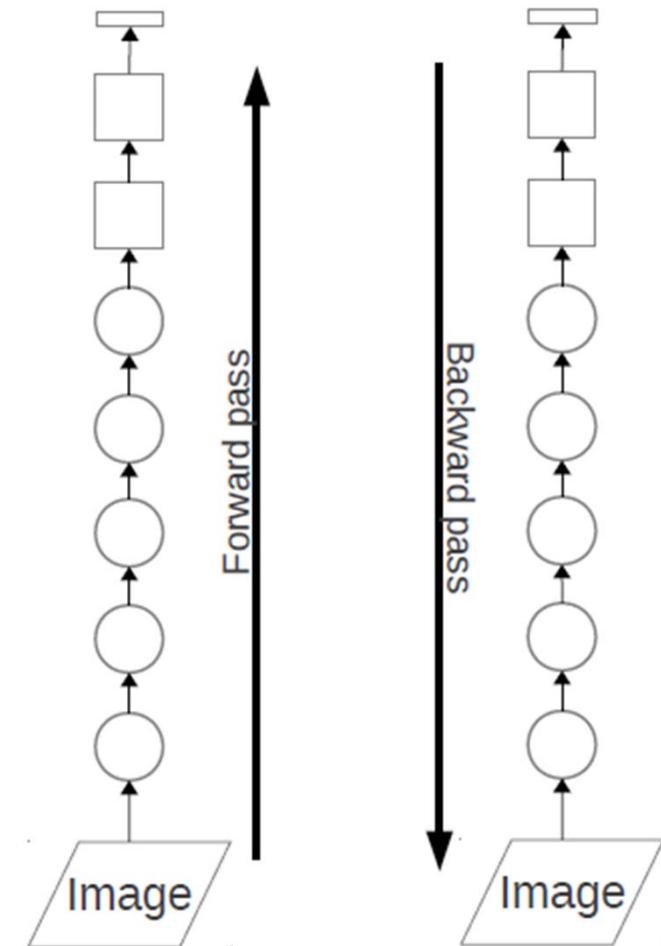
- Entrenamiento:
  - 2 pasos:
    - Algoritmo de *backpropagation*  
(regla de la cadena)



Local convolutional filters

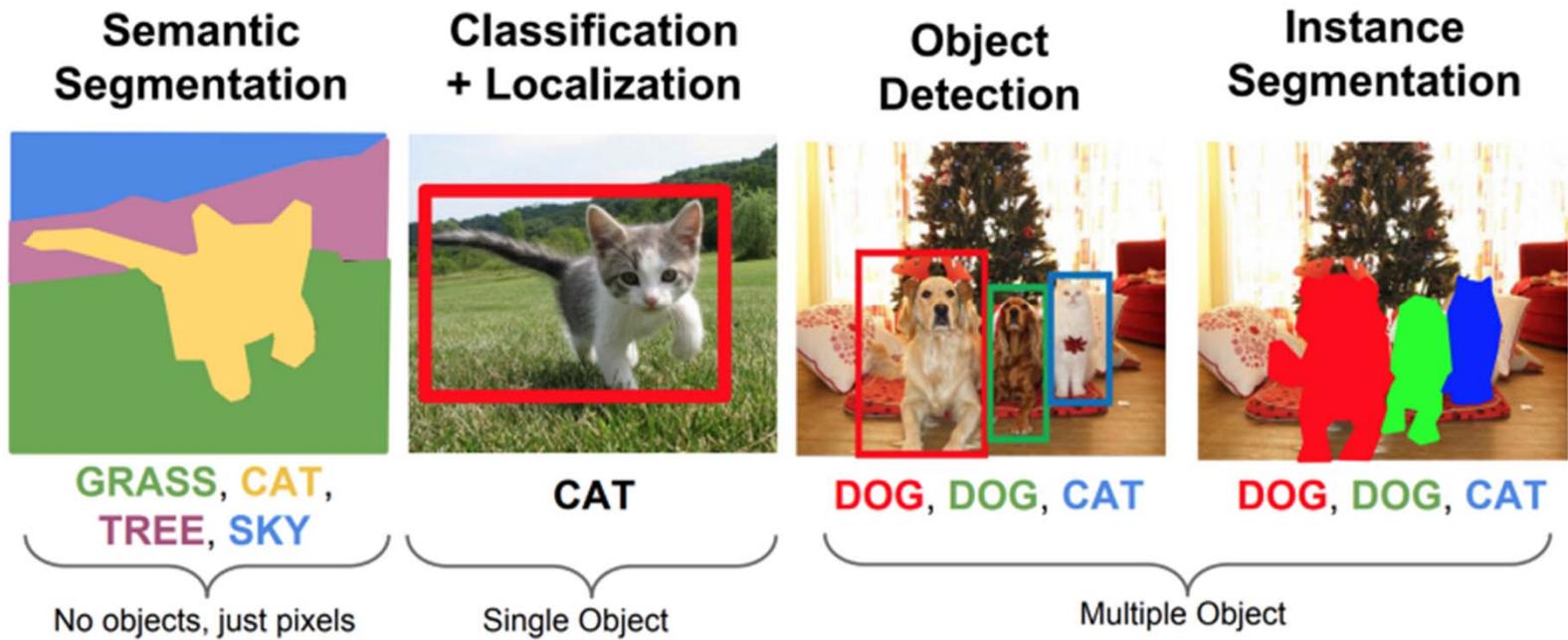


Fully-connected filters



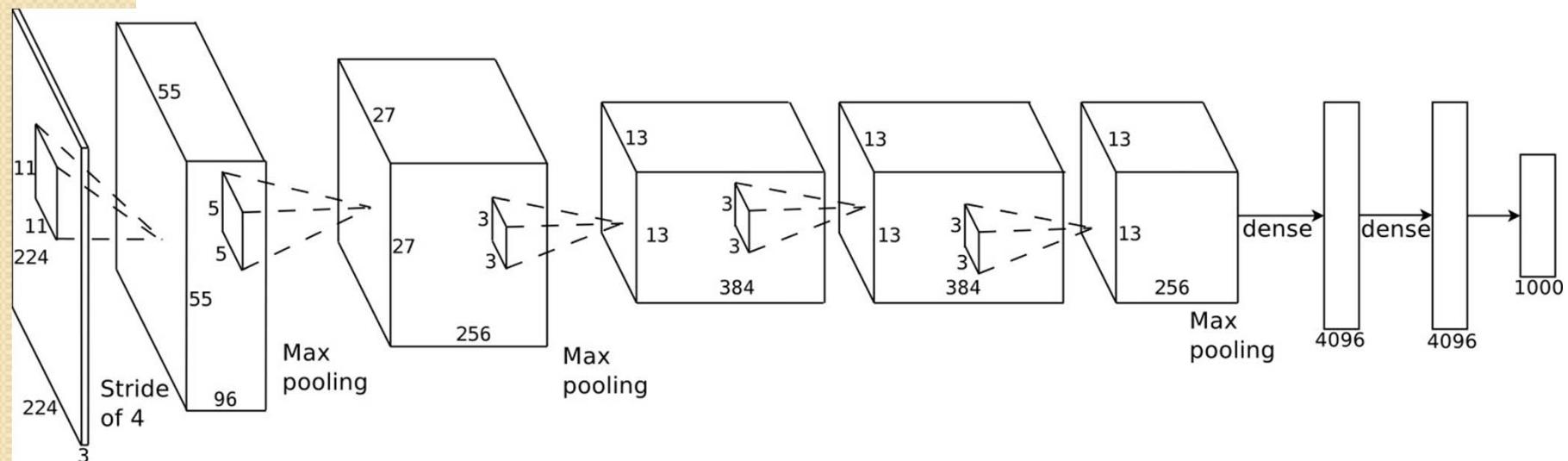
# DEEP LEARNING

- *Convolutional neural networks:*
  - ¿Qué problemas puede resolver?



# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Capas *max-pooling* después de la primera, segunda y quinta capa convolucional



- Número de neuronas en cada capa:
  - 253440, 186624, 64896, 64896, 43264, 4096, 4096, 1000

# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - 96 filtros de bajo nivel aprendidos por la primera capa convolucional



# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Más de 600000 neuronas y ~60M de parámetros como valores reales
    - Sobreentrena muchísimo
      - Por tanto, a partir de imágenes de tamaño 256x256 se tomaron distintos trozos aleatorios de tamaño 224x224
        - Y sus reflejos horizontales





# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Para evaluar una imagen:
      - Se evalúan 5 trozos de tamaño 224x224 y sus reflejos horizontales
        - En las 4 esquinas y en el centro
      - Se toma el promedio de esas predicciones

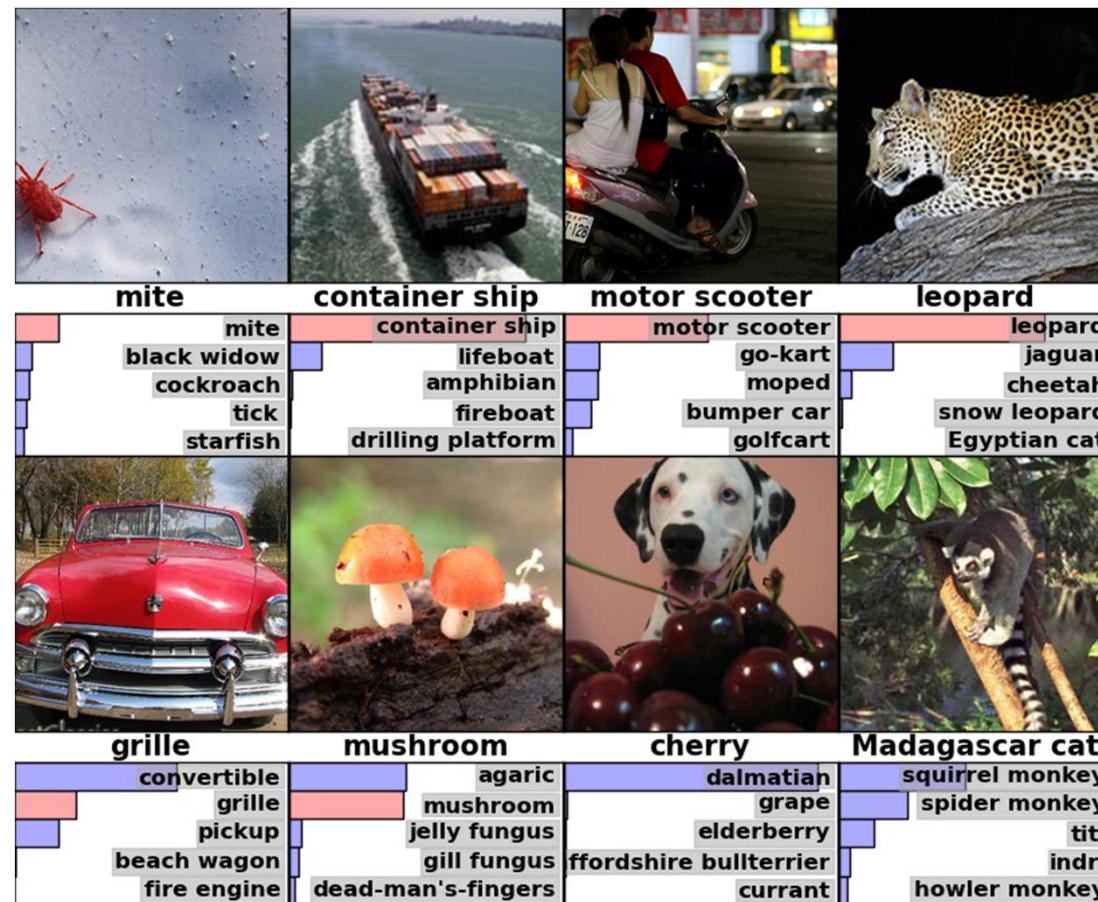


# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Utilizada en la competición ILSVRC-2012
      - Etiquetas de test no publicadas
      - Se muestran aquí resultados en validación
      - Para cada imagen se muestran las 5 etiquetas que este modelo considera las más probables
      - La etiqueta correcta se muestra debajo de cada imagen
      - Entre las etiquetas más probables, la correcta se muestra en una barra rosada

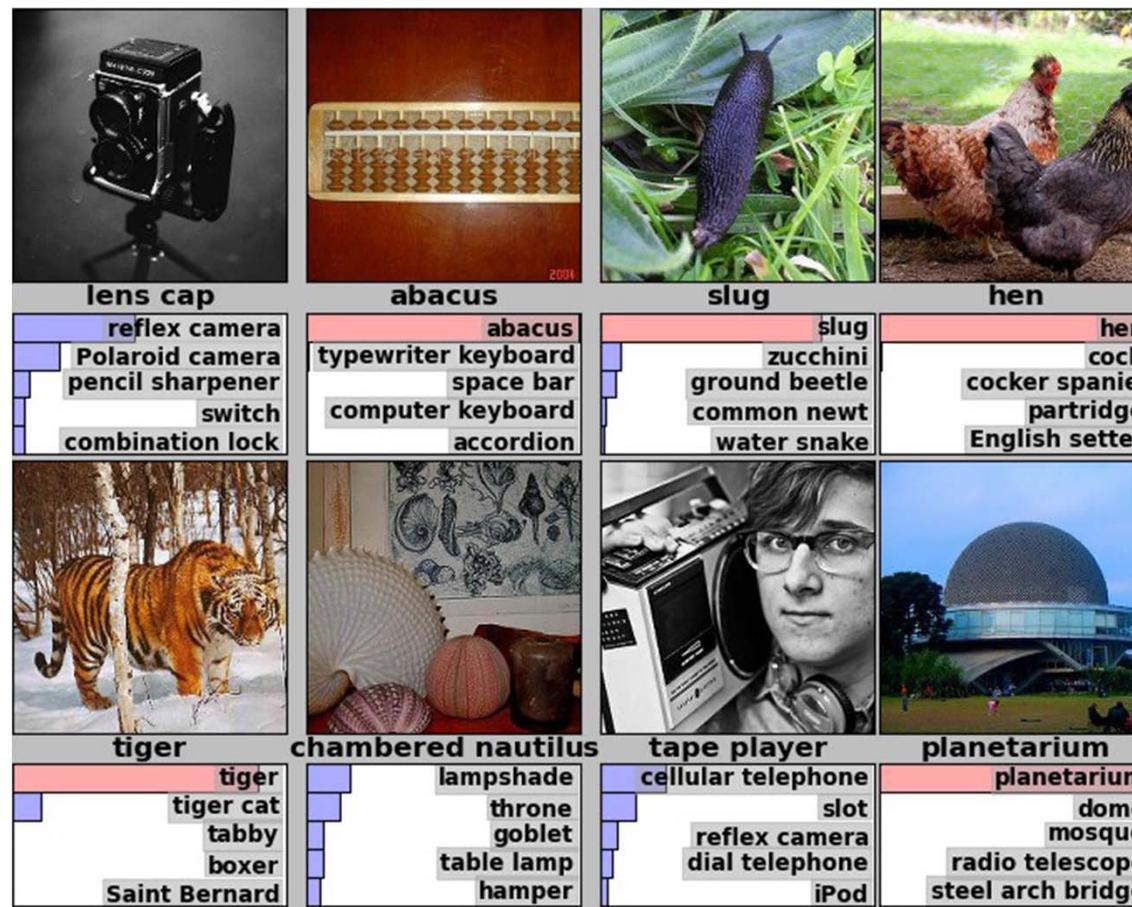
# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:



# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:



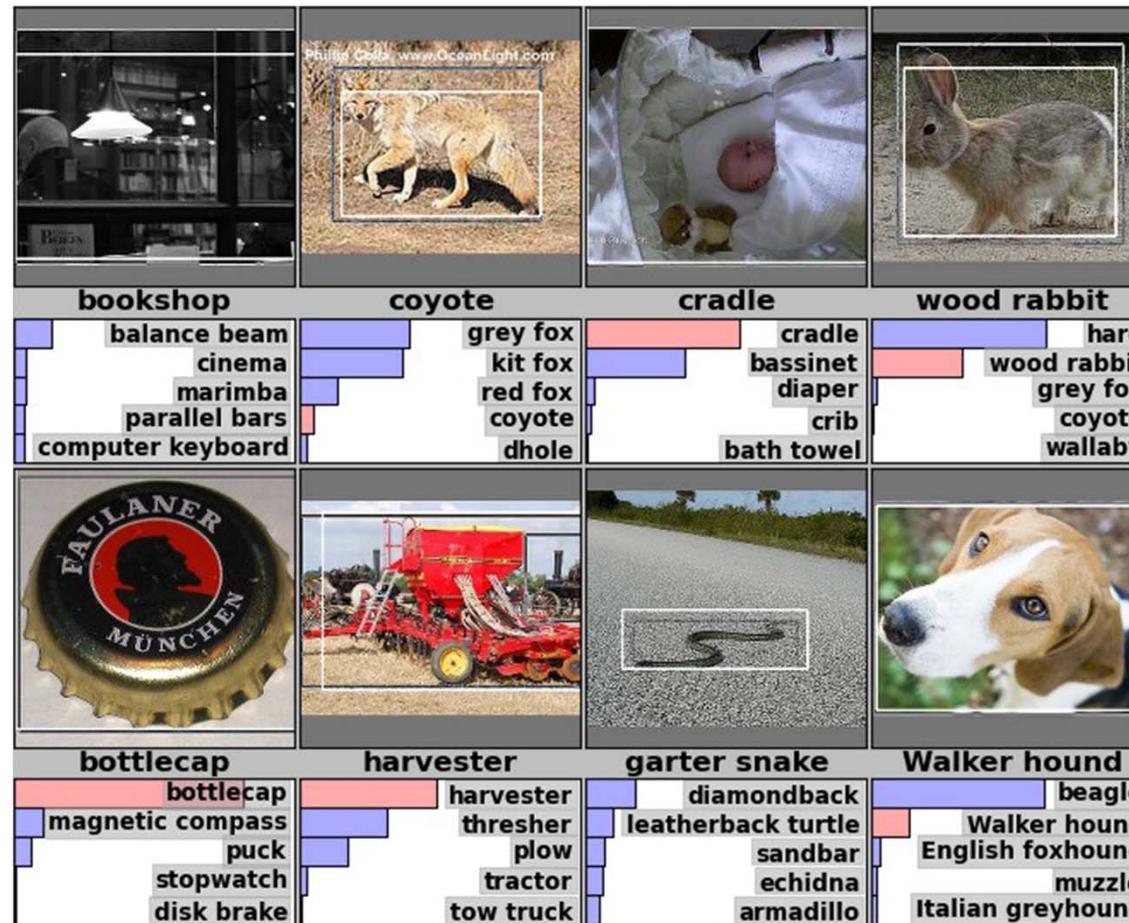
# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:



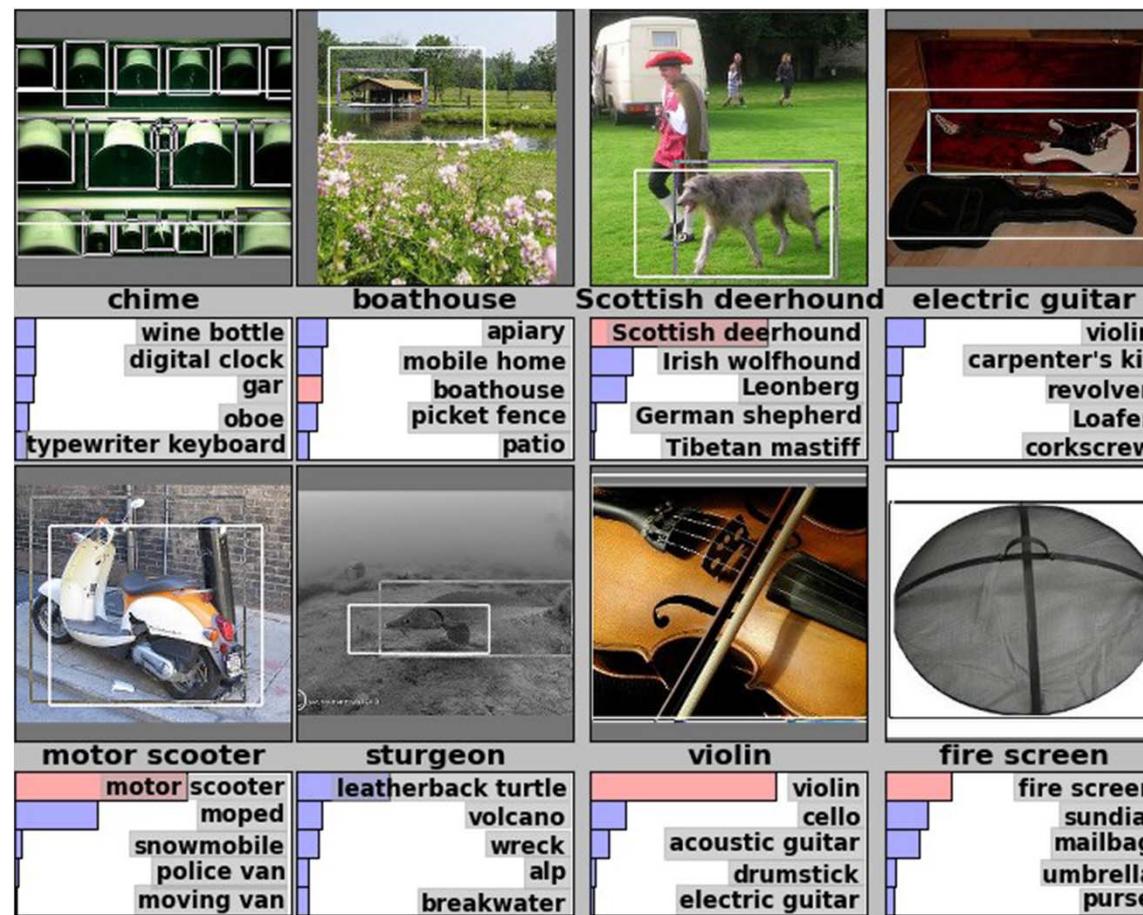
# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo: Detección en validación:



# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo: Detección en validación:



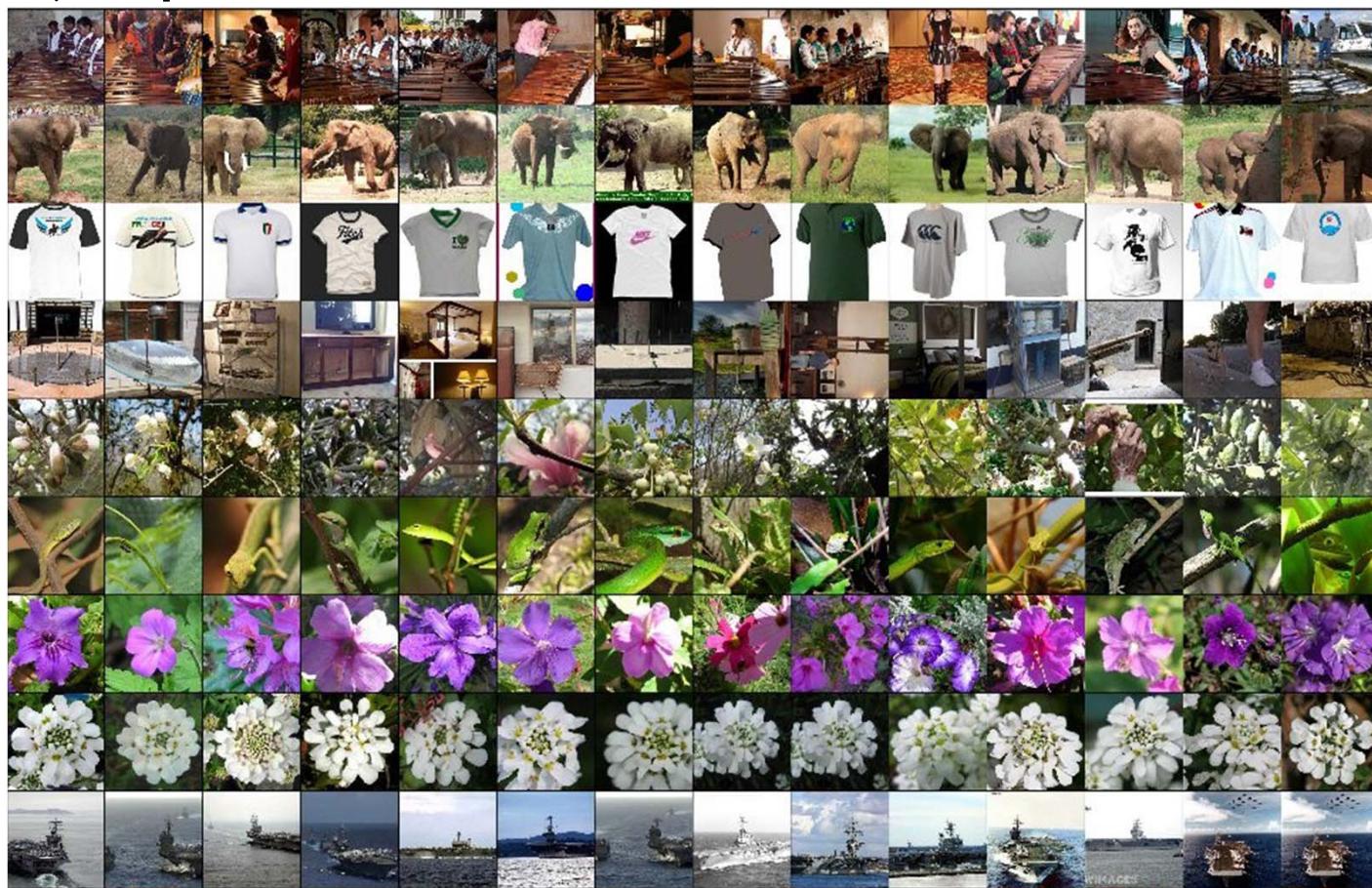


# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Primera columna:
      - Imágenes del conjunto de **test** de ILSVRC-2010
    - Resto de columnas:
      - Imágenes del conjunto de entrenamiento que producen vectores de características en la última capa oculta con la menor distancia euclídea al vector de características de la imagen de test correspondiente (primera columna)

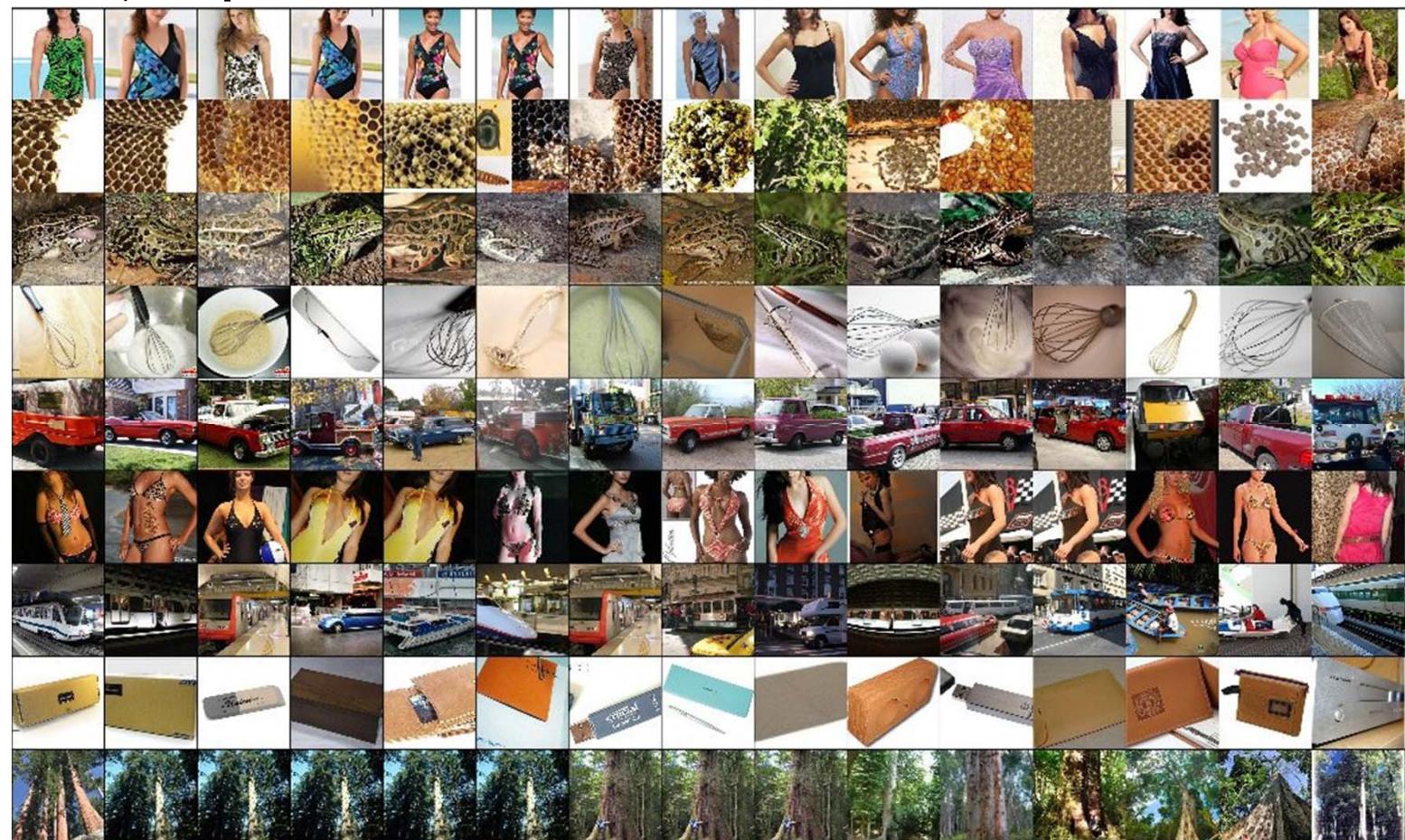
# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:



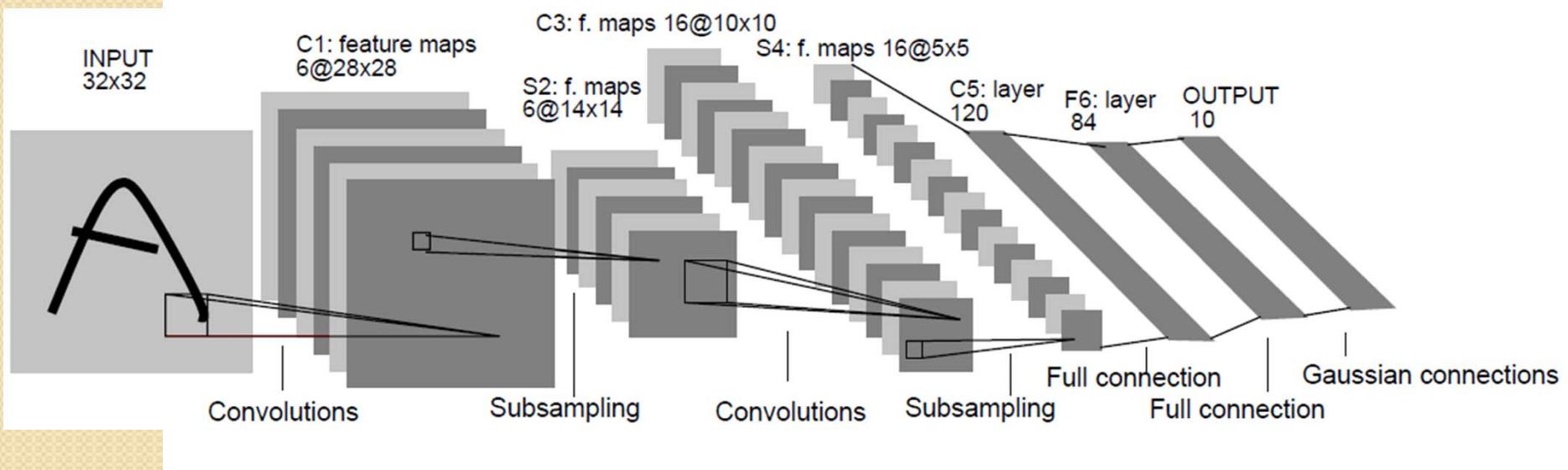
# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:



# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - LeNet-5
      - Diseñada para reconocimiento de caracteres escritos a mano o mecanografiados
      - <http://yann.lecun.com/exdb/lenet/>



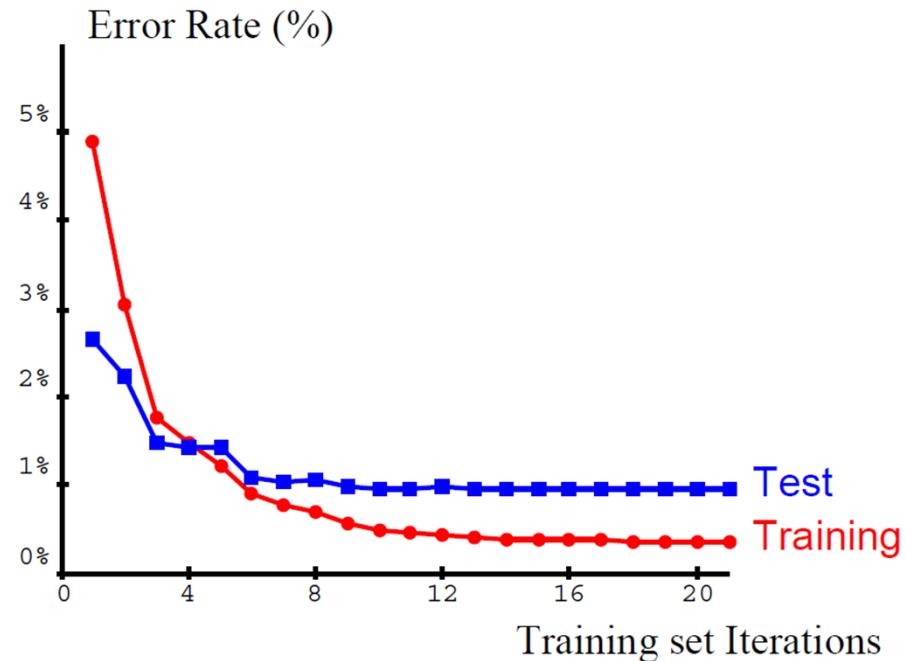
# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Ejemplos tomados de la base de datos utilizada:

3	6	8	1	7	9	6	6	9	1
6	7	5	7	8	6	3	4	8	5
2	1	7	9	7	1	2	8	4	6
4	8	1	9	0	1	8	8	9	4
7	6	1	8	6	4	1	5	6	0
7	5	9	2	6	5	8	1	9	7
2	2	2	2	3	4	4	8	0	
0	2	3	8	0	7	3	8	5	7
0	1	4	6	4	6	0	2	4	3
7	1	2	8	1	6	9	8	6	1

# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Resultados:
      - 60000 patrones en el conjunto de entrenamiento
      - Se alcanzó convergencia después de 10-12 pasos del conjunto de entrenamiento

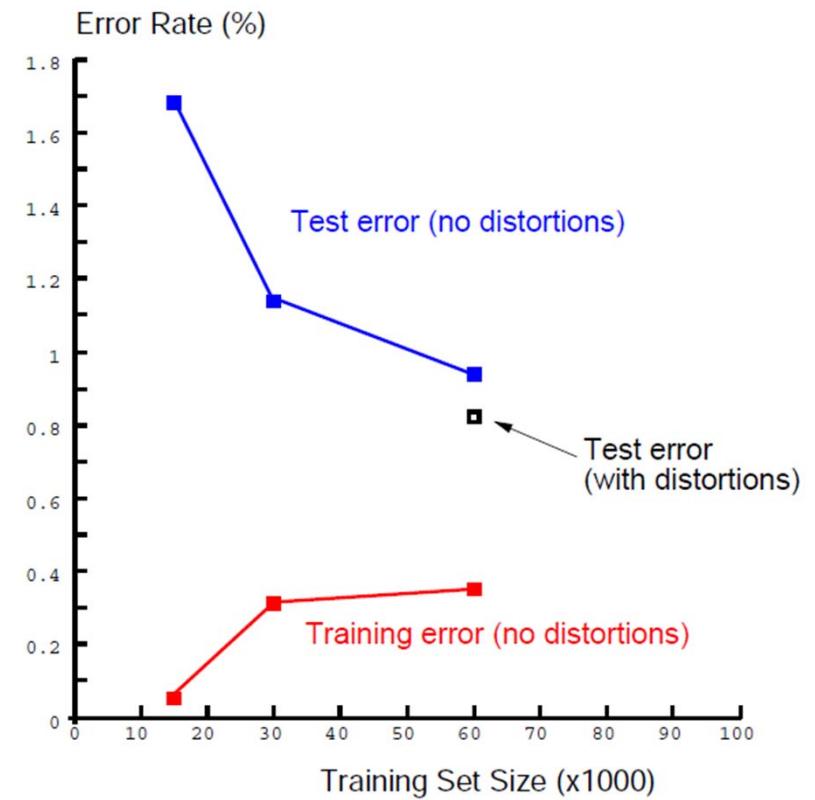


# DEEP LEARNING

- *Convolutional neural networks:*
    - Ejemplo:
      - Imágenes distorsionadas utilizadas
        - Ejemplos de distorsiones de 10 imágenes:

# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Resultados con distintos tamaños del conjunto de entrenamiento y con imágenes distorsionadas



# DEEP LEARNING

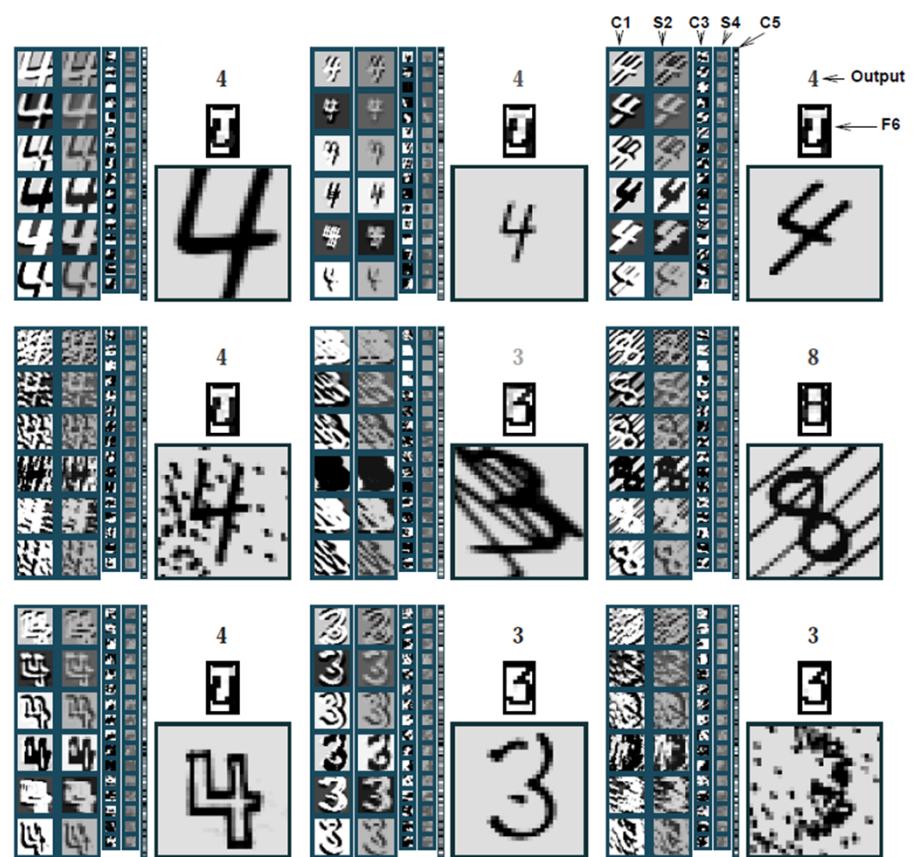
- *Convolutional neural networks:*
    - Ejemplo:
      - 82 patrones incorrectamente clasificados:

- Bajo cada imagen:
    - Izq: Correcta
    - Dcha: Salida RNA

# DEEP LEARNING

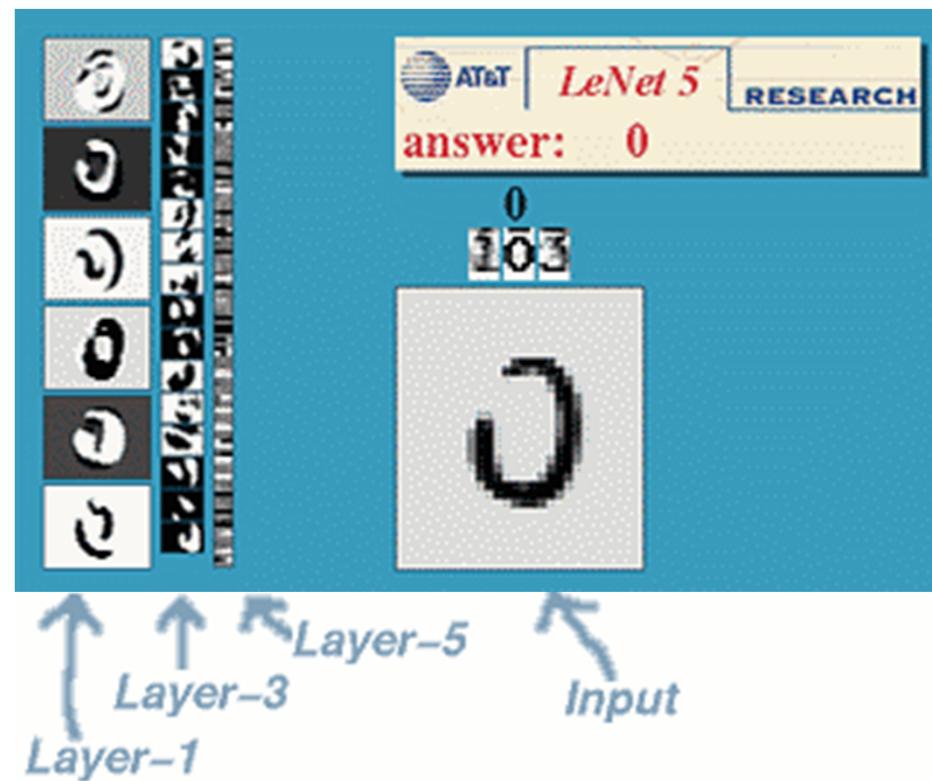
- *Convolutional neural networks:*

- Ejemplo:
  - Ejemplos de patrones inusuales, distorsionados, o con ruido correctamente clasificados



# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Funcionamiento:



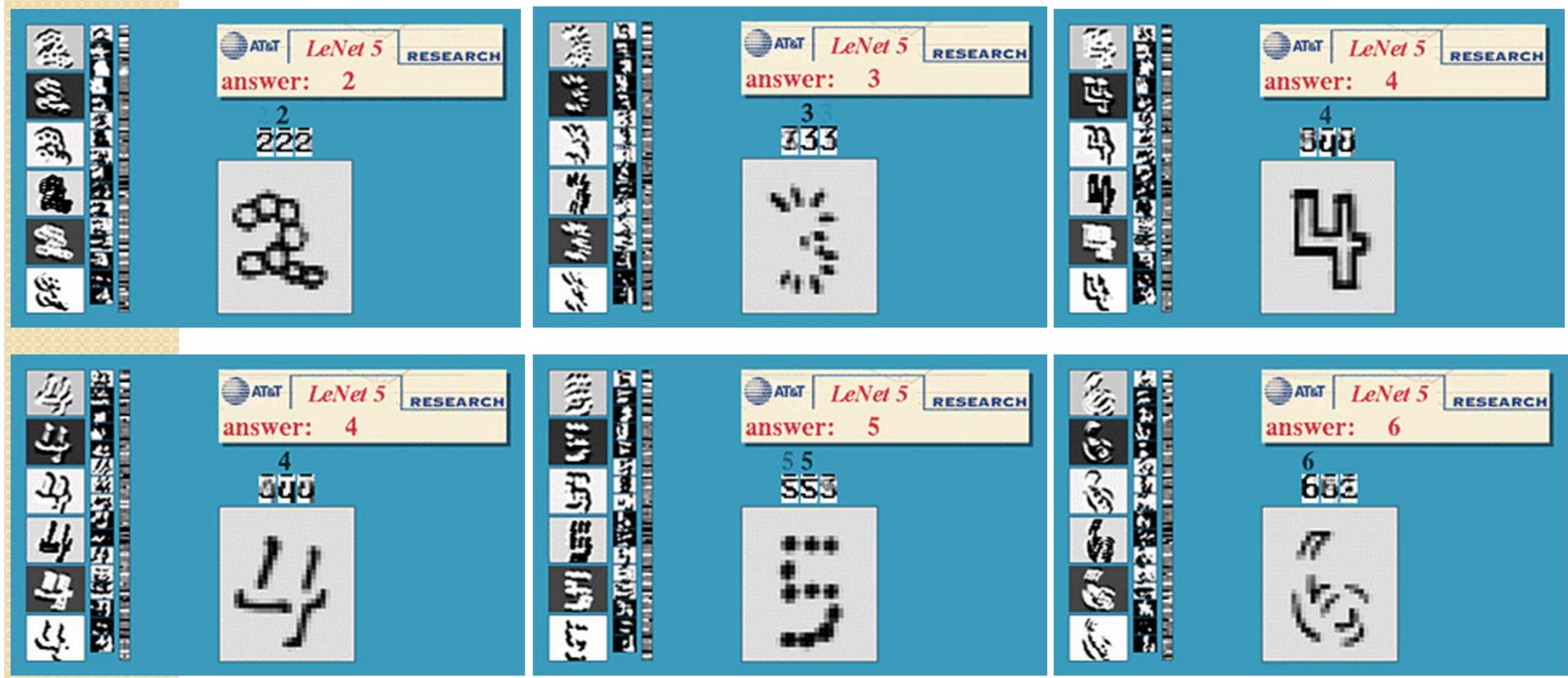
# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Funcionamiento: Estilos inusuales:



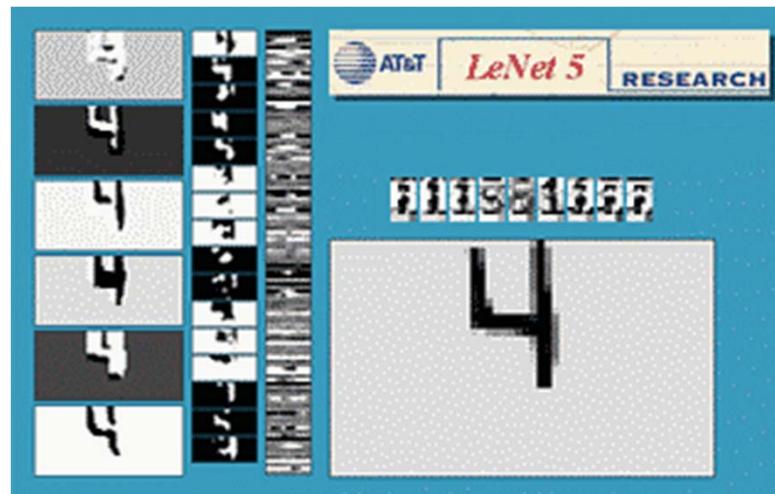
# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Funcionamiento: Entradas extrañas:



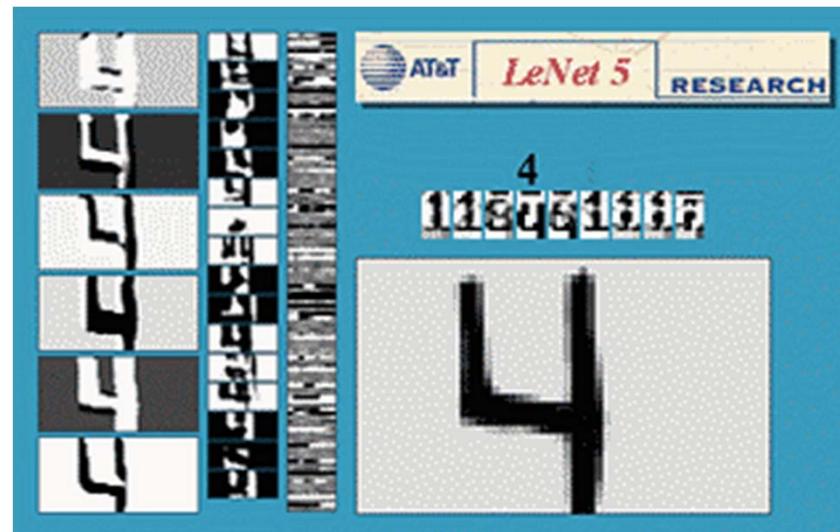
# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Funcionamiento: Movimiento vertical:



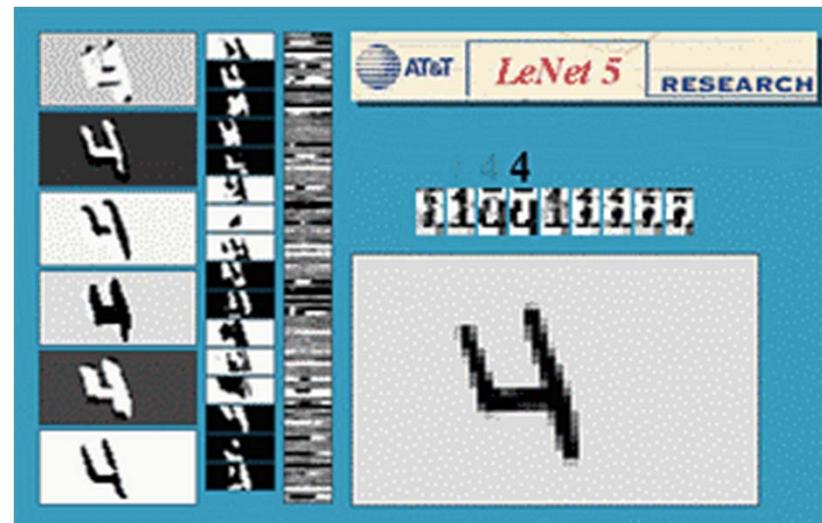
# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Funcionamiento: Escalado:



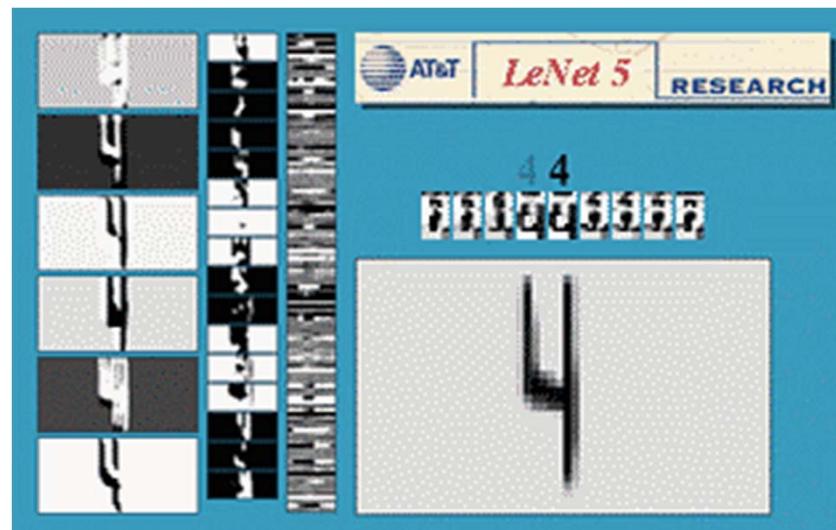
# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Funcionamiento: Rotación:



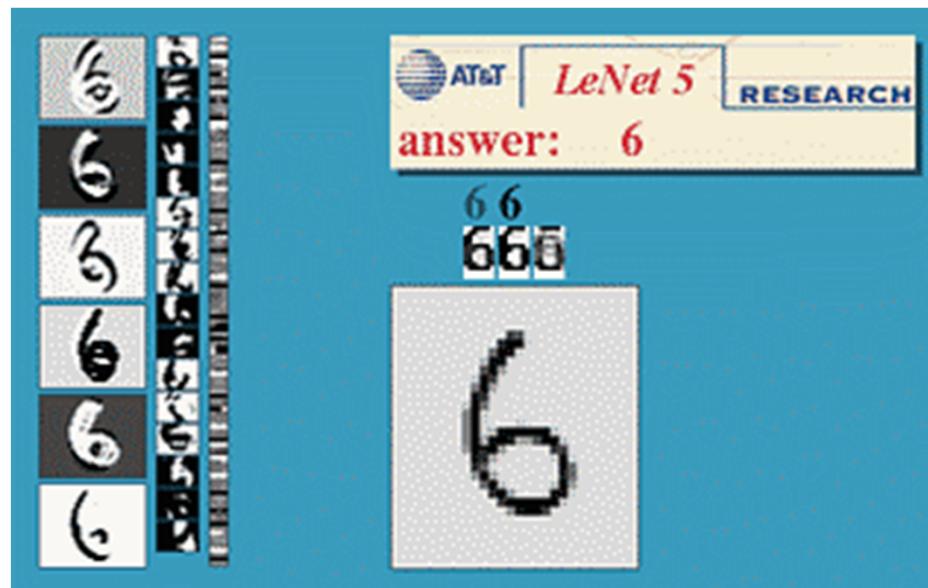
# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Funcionamiento: Deformación:



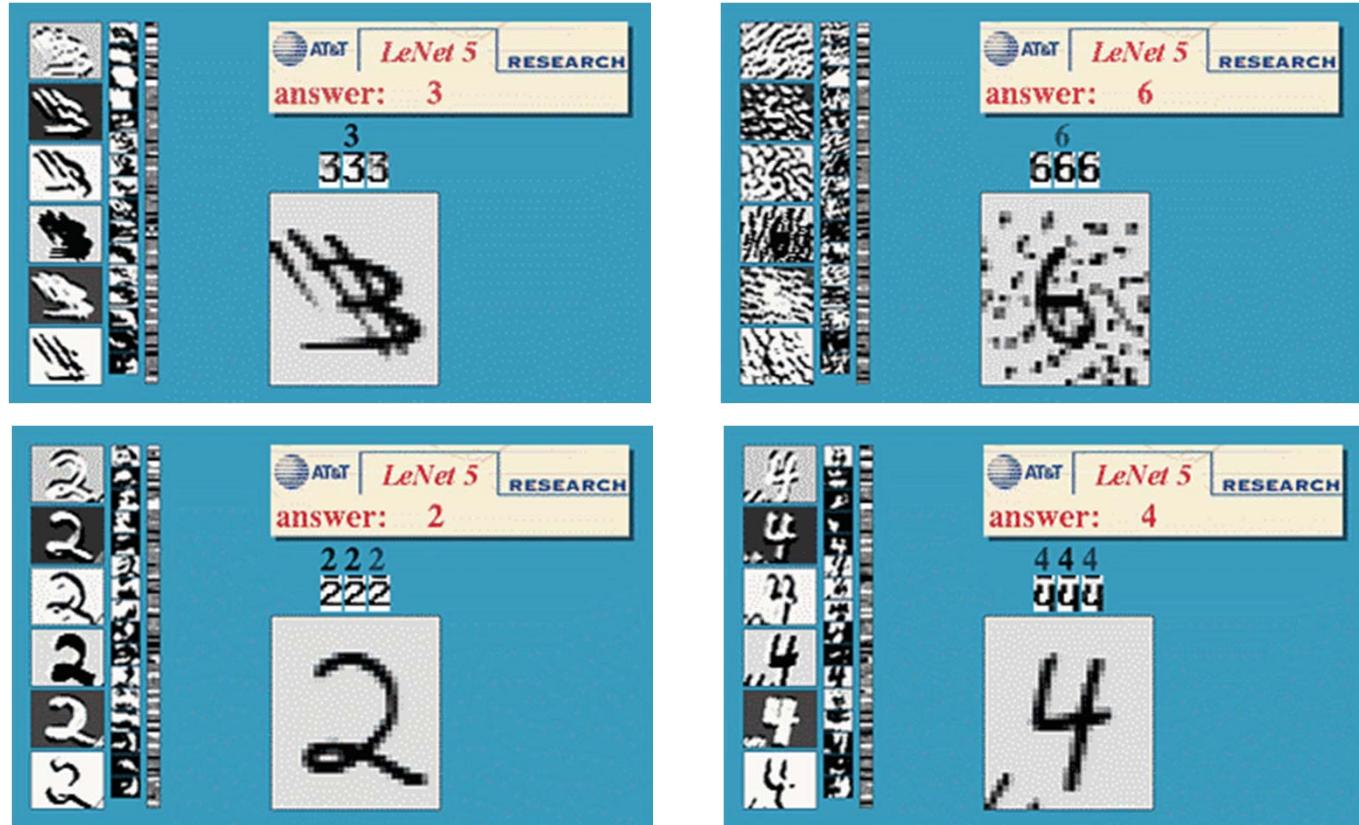
# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Funcionamiento: Distintos trazos:



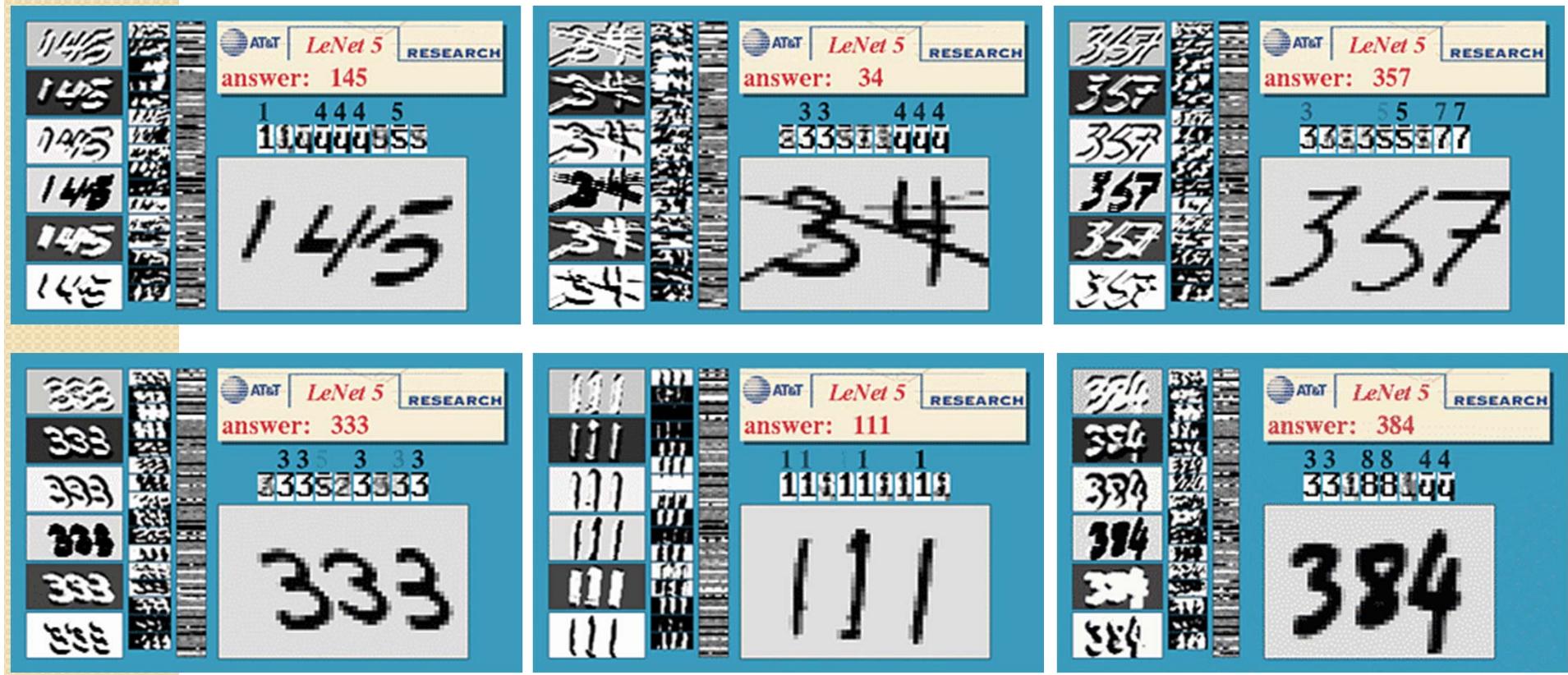
# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Funcionamiento: Ruido:



# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Funcionamiento: Múltiples caracteres:



# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Funcionamiento: Casos complejos:





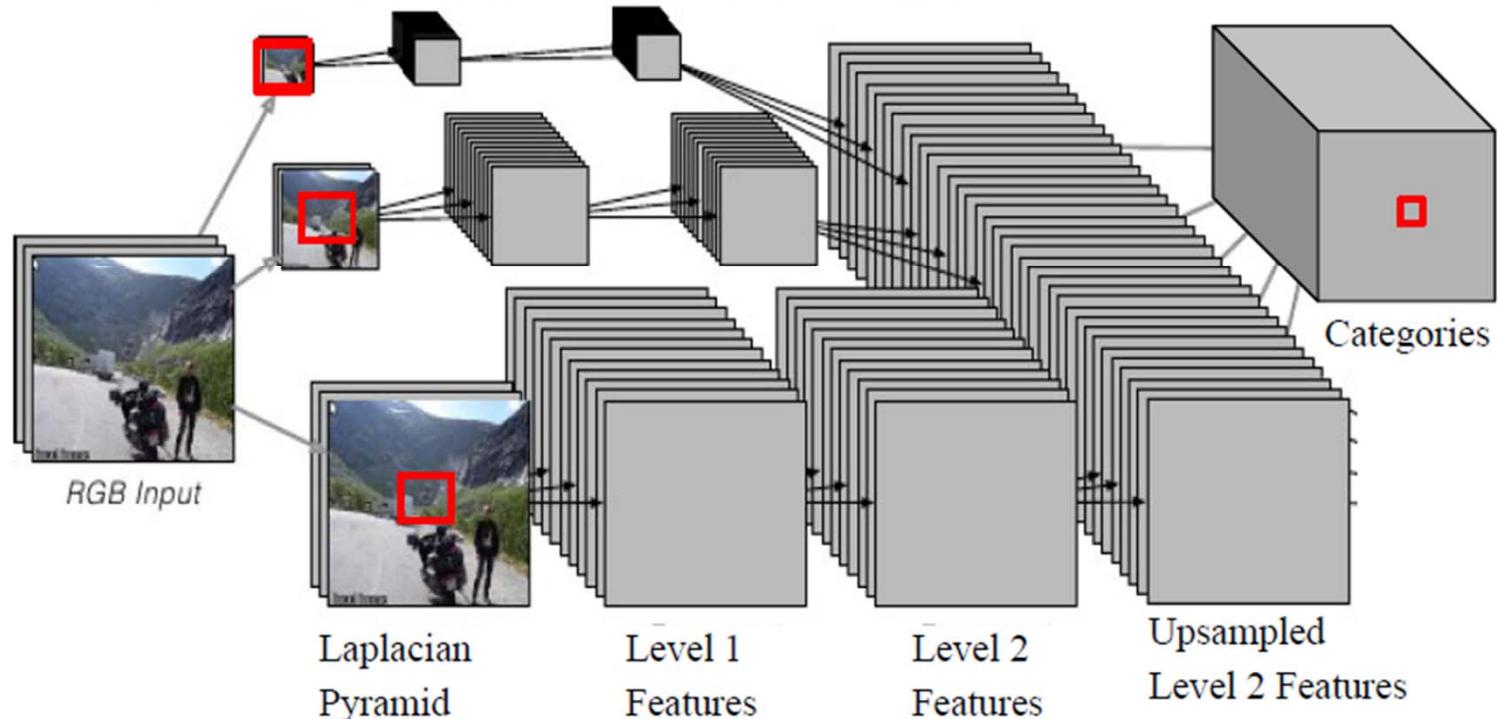
# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Etiquetado semántico:
      - Etiquetar cada pixel al objeto al que pertenece
      - Ayudaría a identificar obstáculos, objetivos, sitios de aterrizaje, zonas peligrosas, etc.
      - Ayudaría a trazar mapas
      - Farabet et al. ICML 2012, PAMI 2013

# DEEP LEARNING

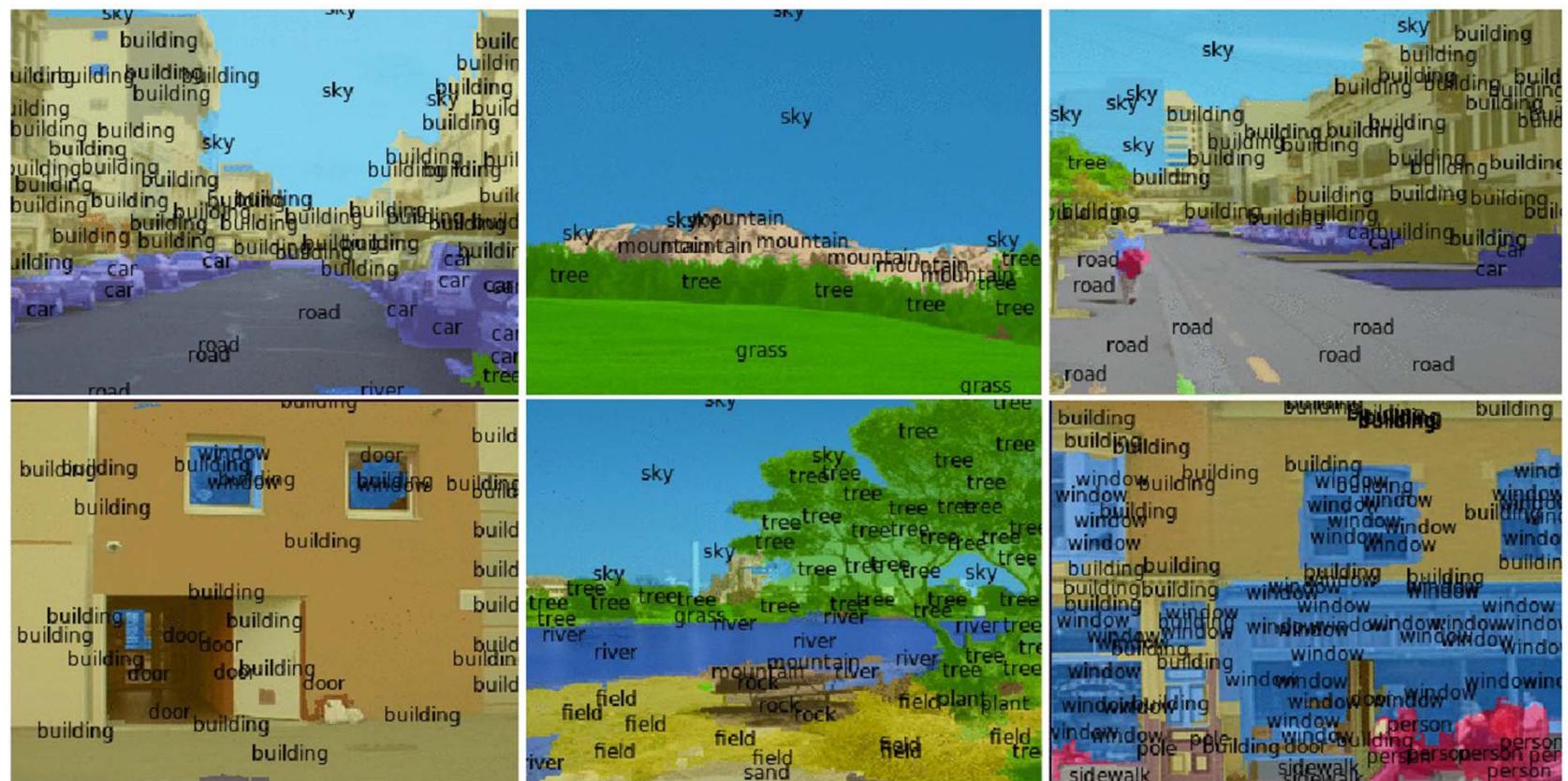
- *Convolutional neural networks:*
  - Ejemplo:

- [46x46 window at full rez; 92x92 at ½ rez; 184x184 at ¼ rez]
- [7x7conv]->[2x2pool]->[7x7conv]->[2x2pool]->[7x7conv]->
- Trained supervised on fully-labeled images



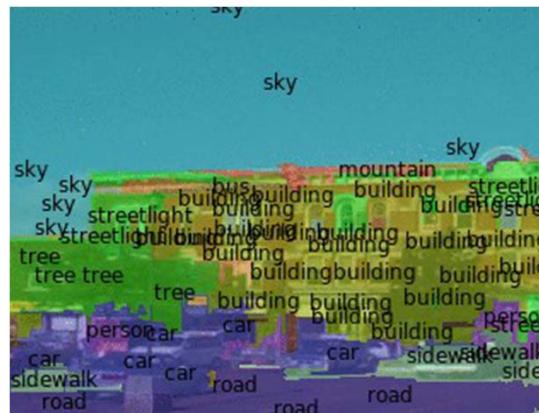
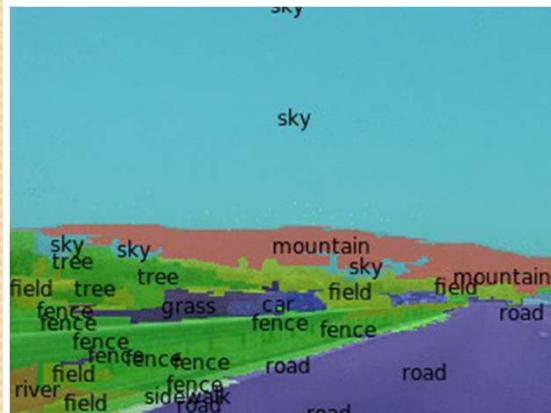
# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Base de datos SIFT-Flow:



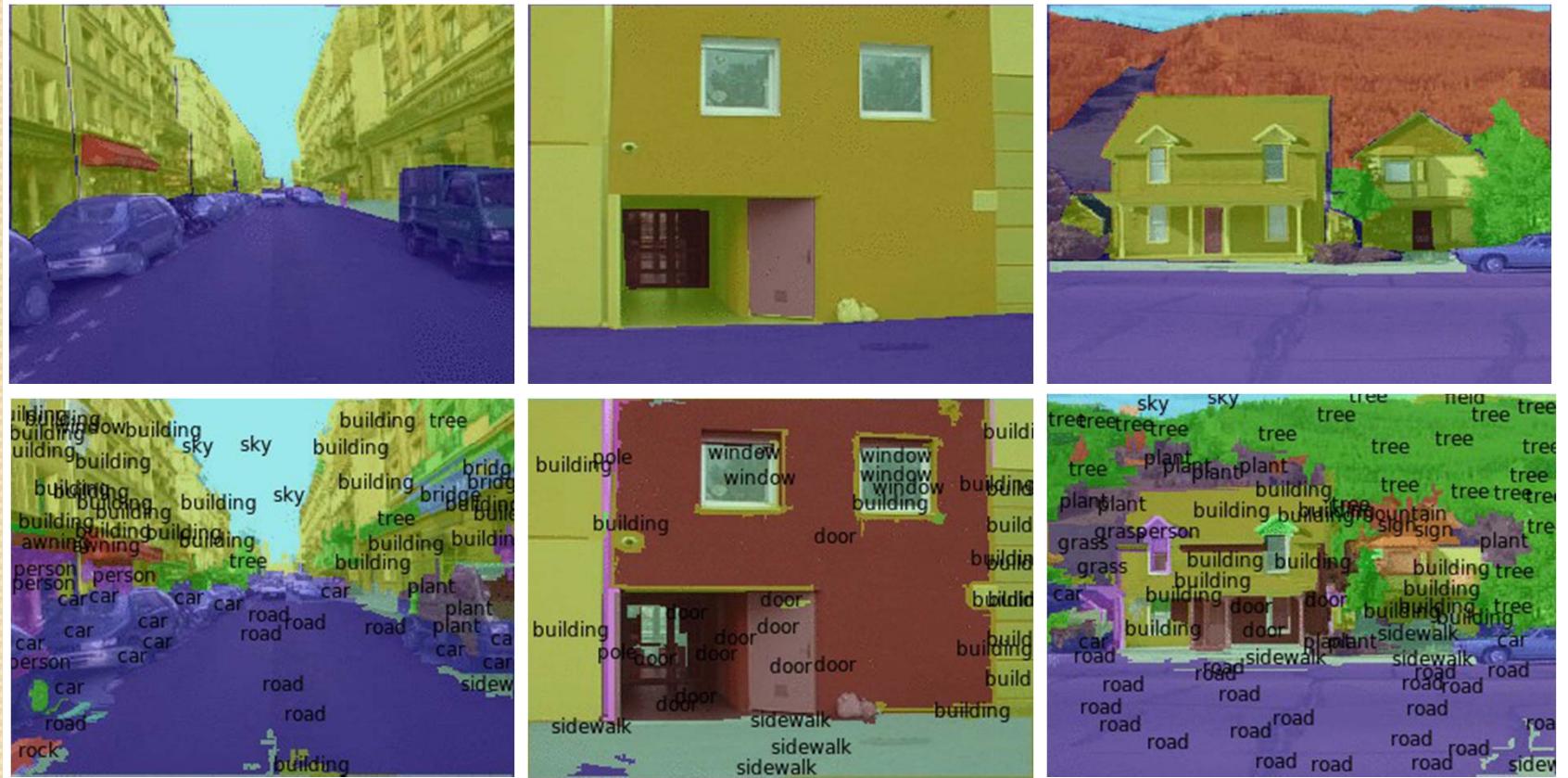
# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Etiquetado de escenas:



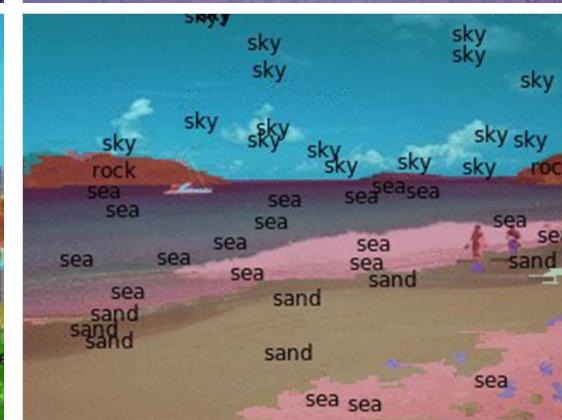
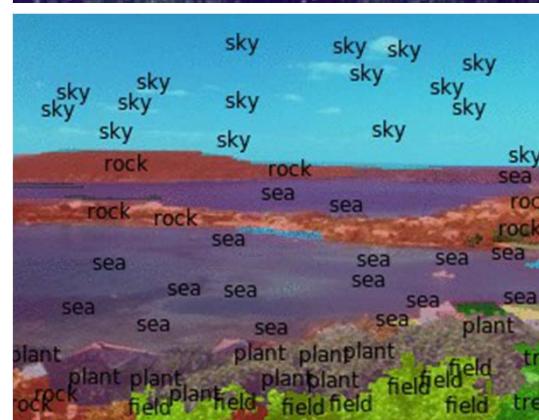
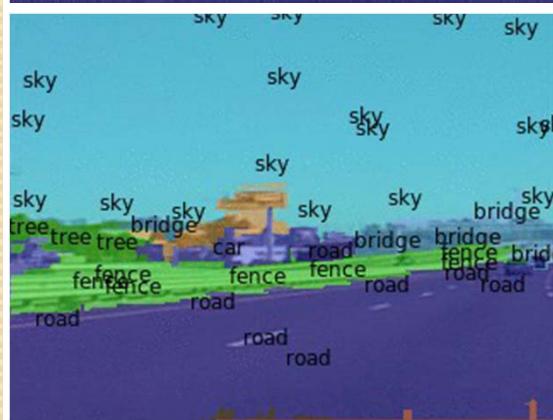
# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Etiquetado de escenas:



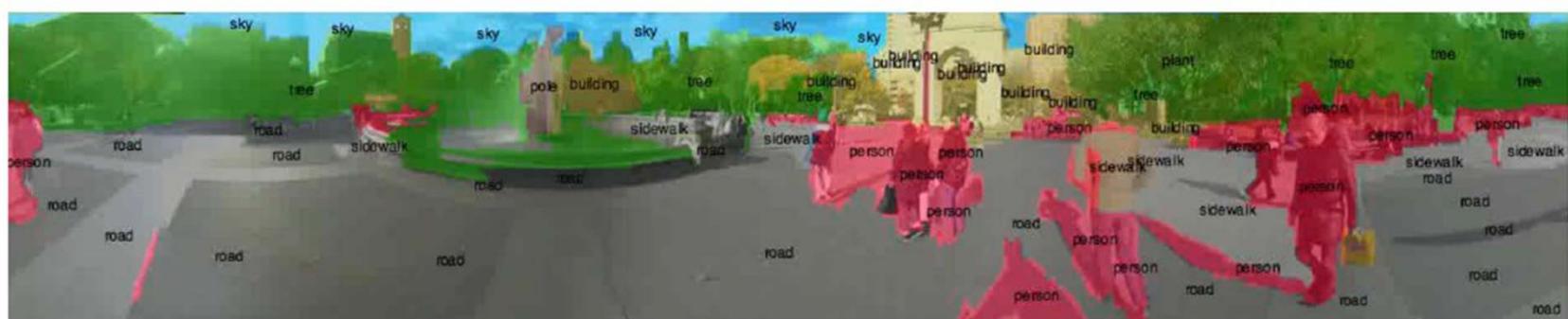
# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Etiquetado de escenas:



# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Etiquetado de escenas:



# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Etiquetado de escenas:



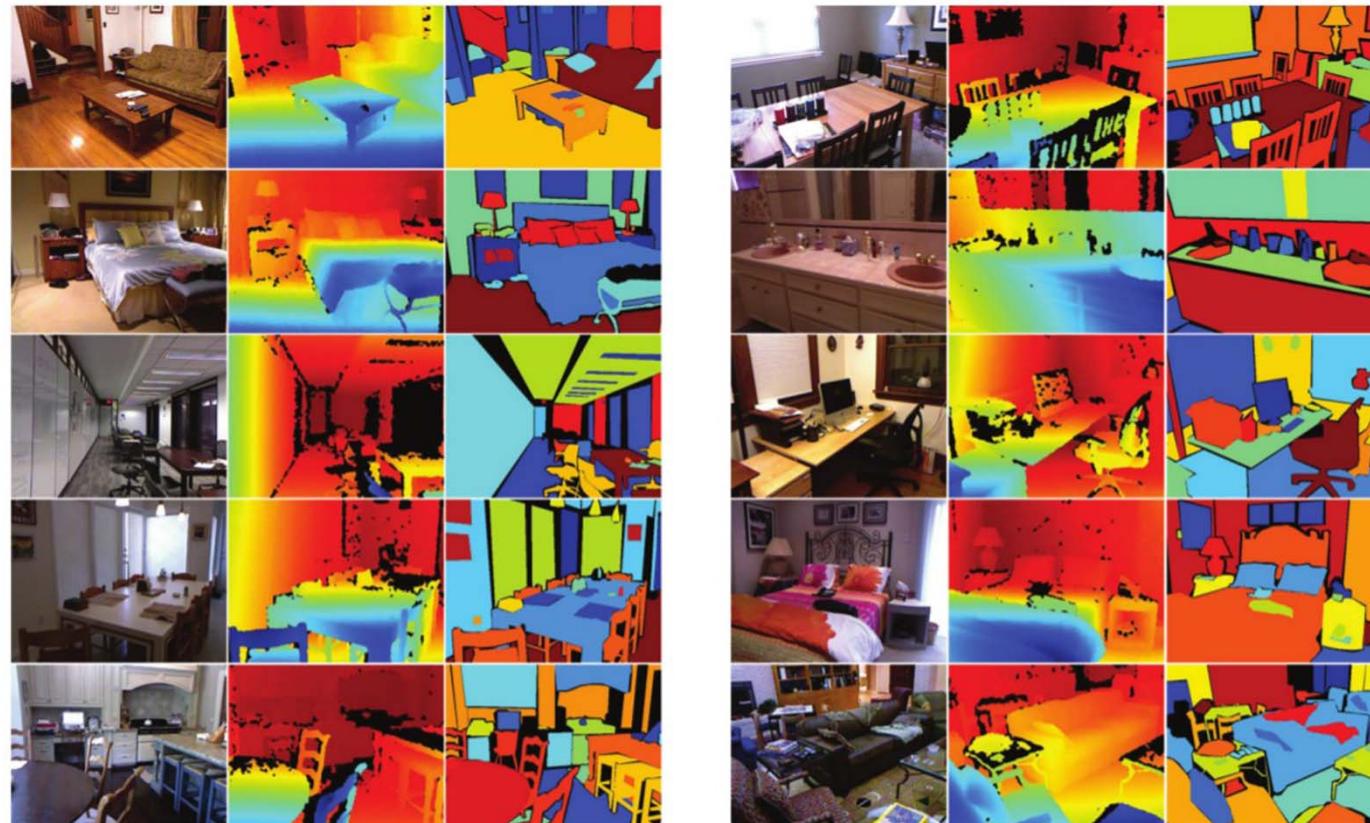


# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Etiquetado de escenas:
      - Escenas en interiores:
        - 407024 imágenes de apartamentos en RGB-D
        - 94 categorías de objetos
        - Silberman et al. 2012
        - Couprie, Farabet, Najman, LeCun ICLR 2013, ICIP 2013

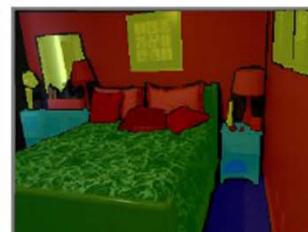
# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Etiquetado de escenas: Imágenes utilizadas:



# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Etiquetado de escenas: Resultados:



Ground truths



Our results

wall  
bed

books  
ceiling

chair  
floor

furniture  
pict./deco

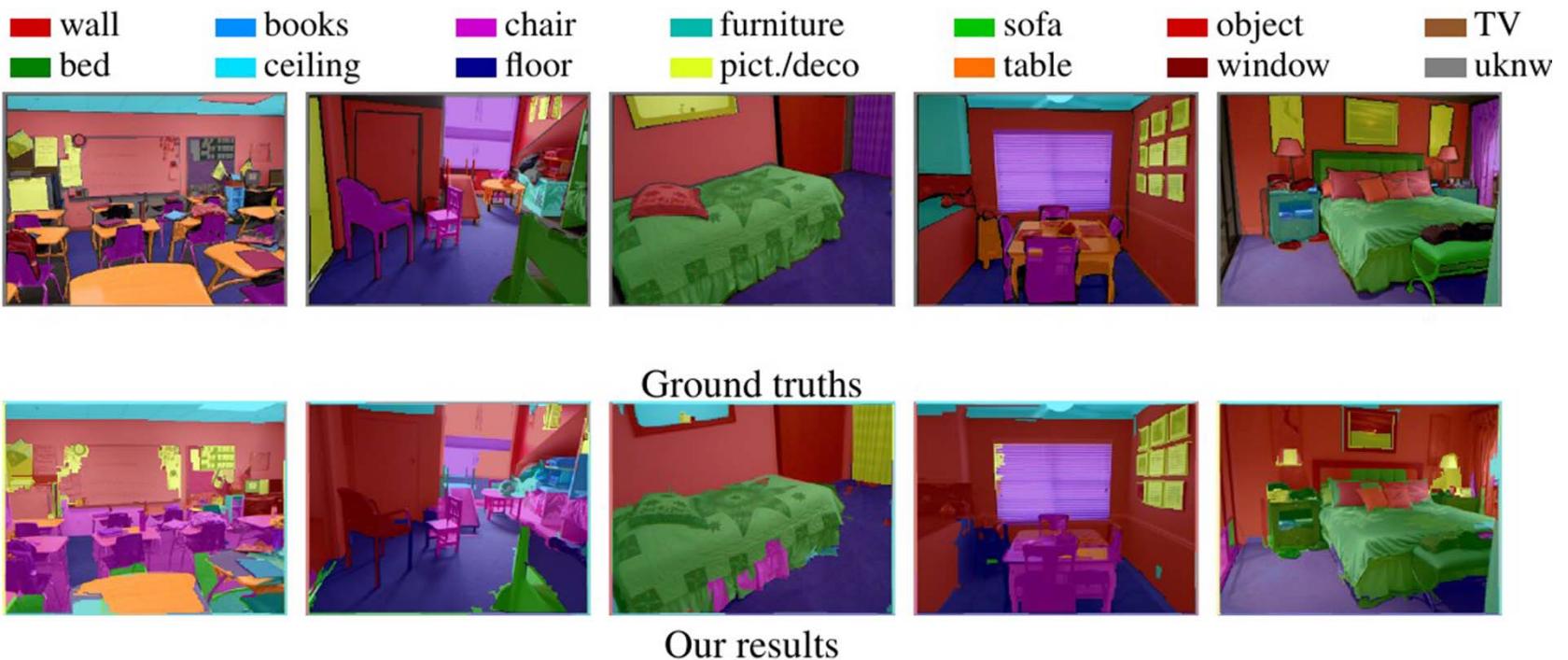
sofa  
table

object  
window

TV  
unkw

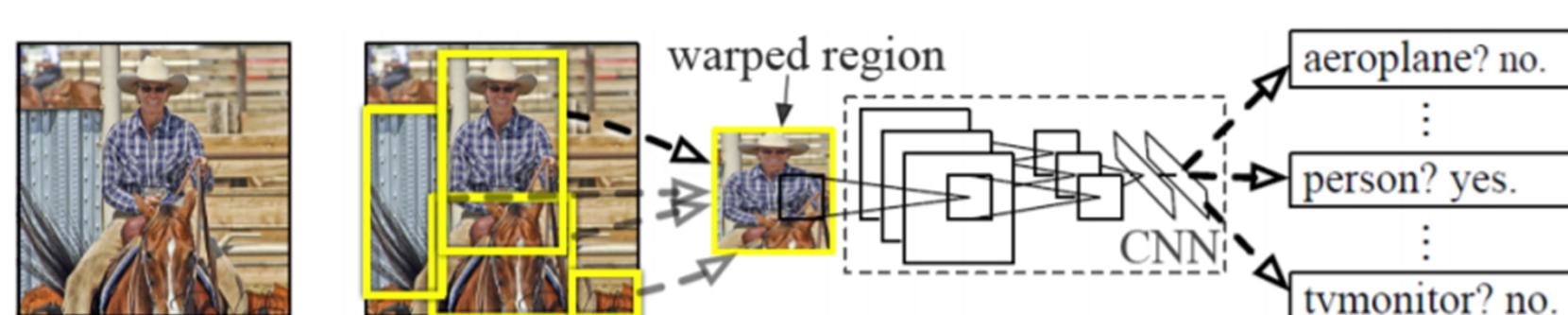
# DEEP LEARNING

- *Convolutional neural networks:*
  - Ejemplo:
    - Etiquetado de escenas: Resultados:



# DEEP LEARNING

- *R-CNN: Region + CNN* □



- *Cómo localizar objetos:*

*Problema de clasificación (tipo de objeto)*

+

*Problema de regresión (sitio del objeto en la imagen)*

# DEEP LEARNING

- *R-CNN: Region + CNN* □

- Proceso:

- 1. Se toma una imagen



Input image

- 2. Se utiliza un algoritmo para seleccionar regiones de interés

- Regiones donde puede haber objetos que detectar
    - Algoritmo más utilizado: Selective Search



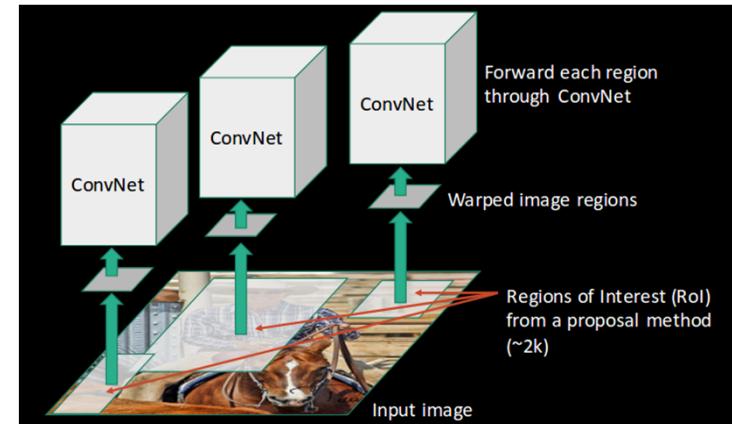
Input image

Regions of Interest (RoI)  
from a proposal method  
(~2k)

# DEEP LEARNING

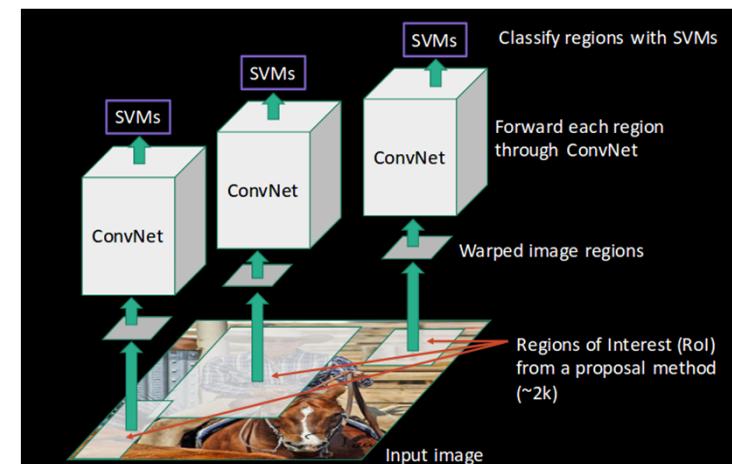
- *R-CNN: Region + CNN* □

3. Se reescalan esas regiones de interés al tamaño de la entrada de la CNN



4. La CNN extrae características de cada región, que se usan como entrada a un clasificador:

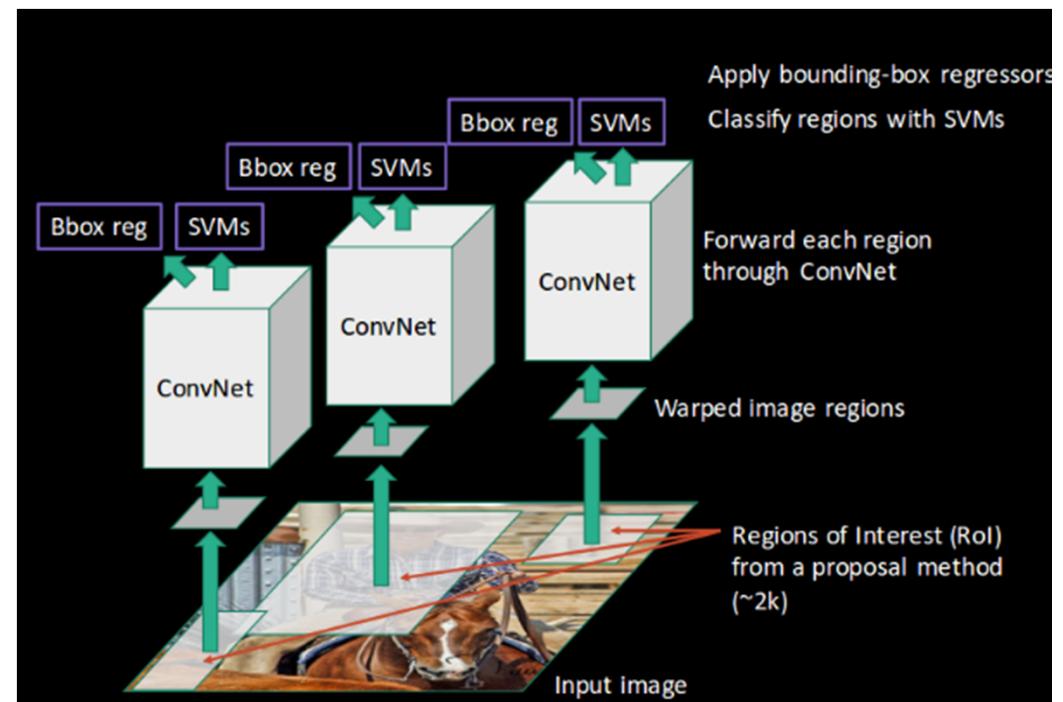
- Perceptrón multicapa
- SVM
- 1 SVM por clase



# DEEP LEARNING

- *R-CNN: Region + CNN* □

5. Una vez resuelto el problema de clasificación, se resuelve el de regresión:
  - Localizar los 4 valores del Bounding Box
  - Regresión lineal

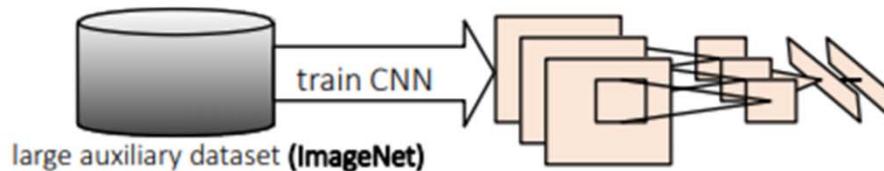


# DEEP LEARNING

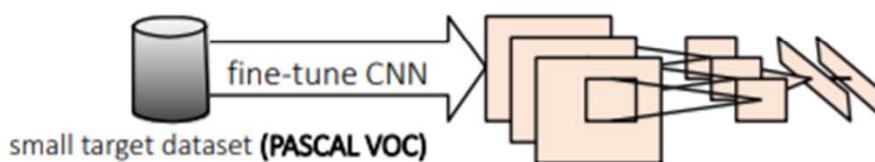
- *R-CNN: Region + CNN* □

## R-CNN: Training

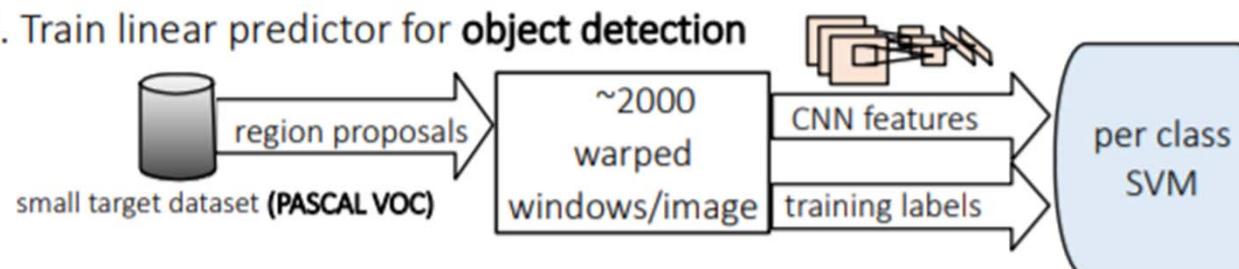
### 1. Pre-train CNN for **image classification**



### 2. Fine-tune CNN for **object detection**



### 3. Train linear predictor for **object detection**



# DEEP LEARNING

- *R-CNN: Region + CNN* □

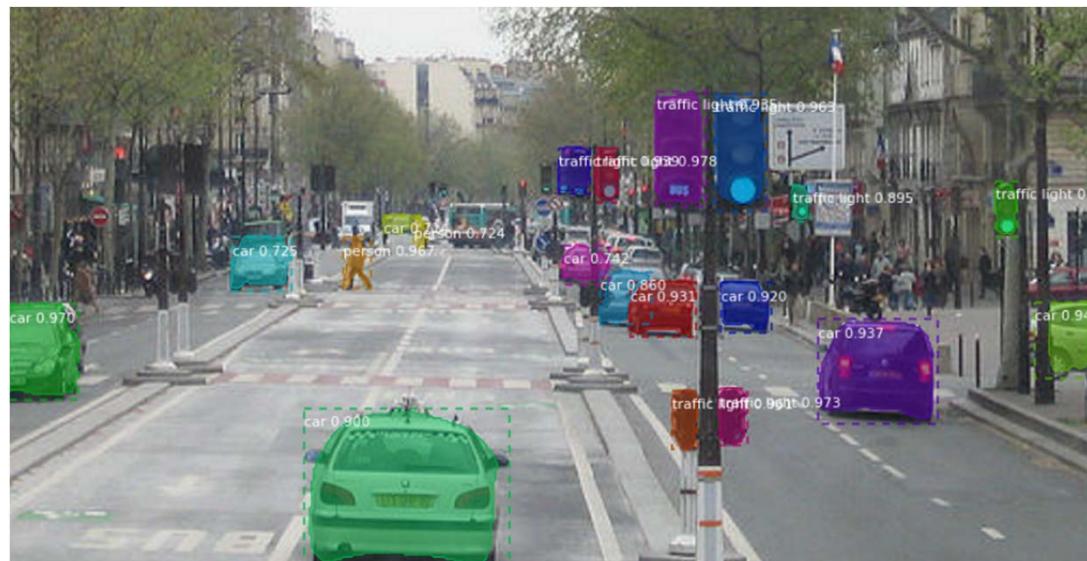
- Ejemplo:

- [https://github.com/karolmajek/Mask□RCNN](https://github.com/karolmajek/MaskRCNN)
    - Otros resultados:



# DEEP LEARNING

- *R-CNN: Region + CNN* □
  - Ejemplo:
    - <https://github.com/karolmajek/MaskRCNN>
    - Resultados:
      - Etiquetado de objetos en imágenes de tráfico en tiempo real:
        - <https://www.youtube.com/watch?v=OOT3UIXZztE>

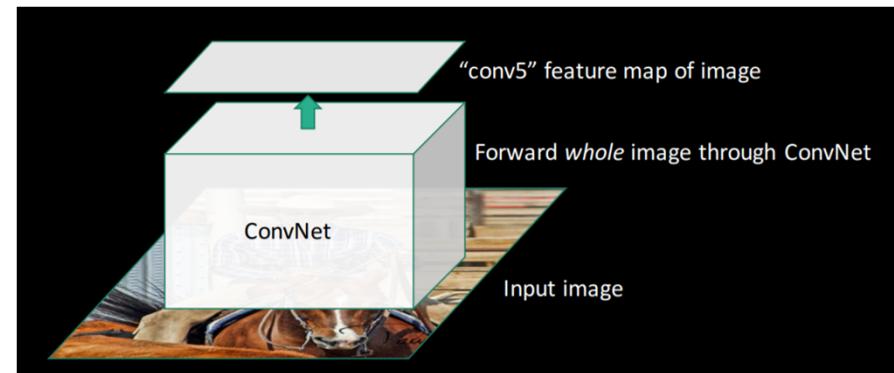


# DEEP LEARNING

- *Fast RCNN:*

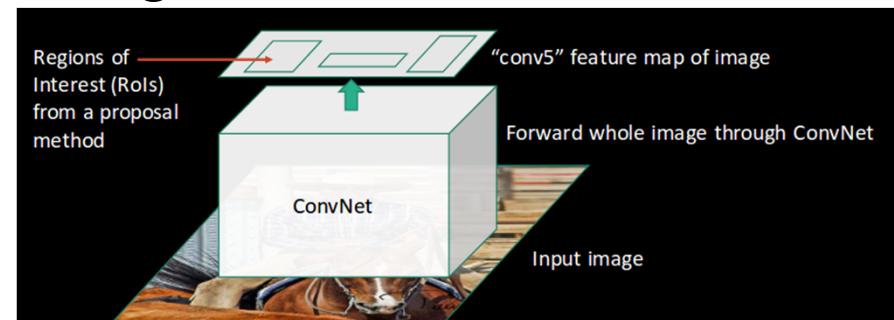
1. Se usa la imagen como entrada de una CNN

- Devuelve una imagen de características



2. Se utiliza un algoritmo para tomar las regiones de interés a partir de esa imagen de características

- *Selective Search*

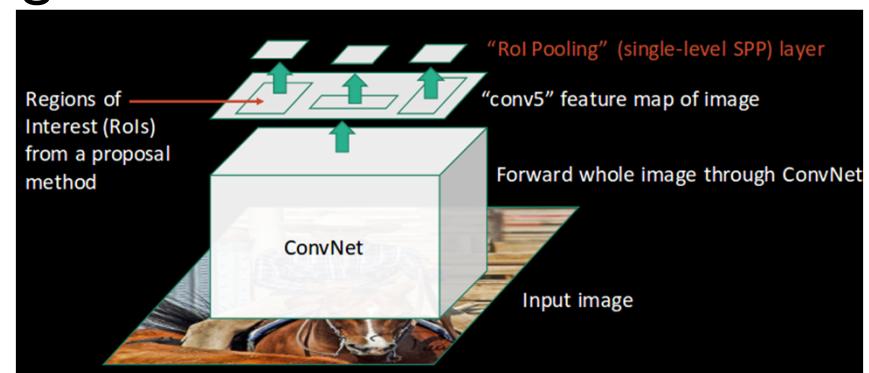


# DEEP LEARNING

- *Fast RCNN:*

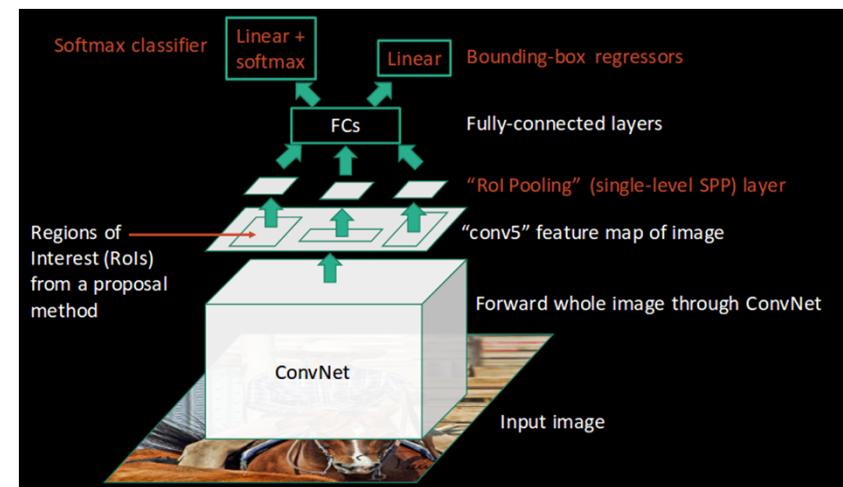
3. Se escalan todas las regiones al mismo tamaño

- Capa «*Roi Pooling*»



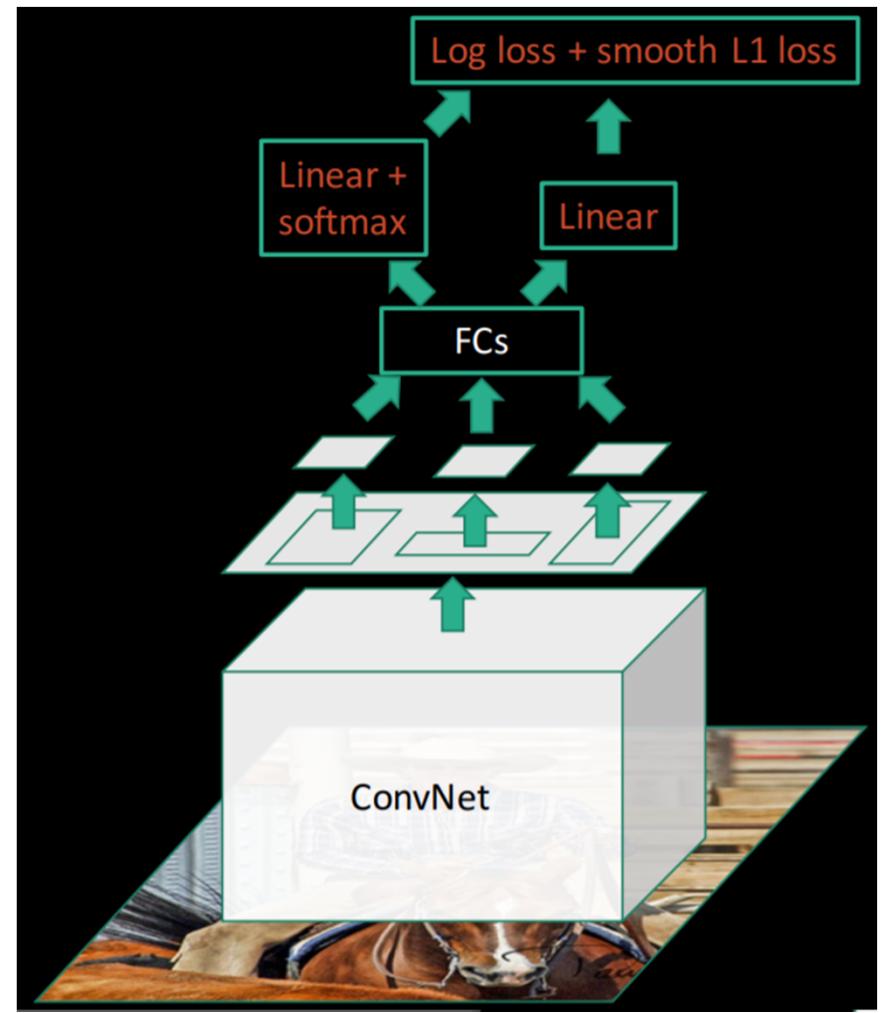
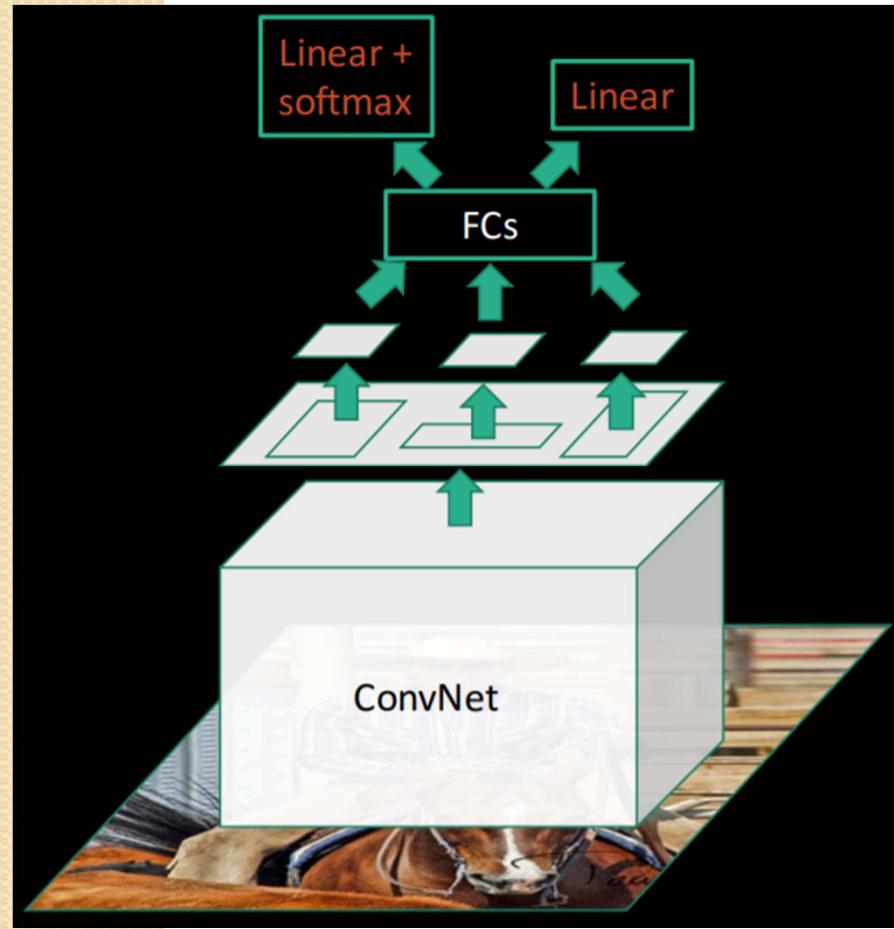
4. Se usan estas regiones como entrada a un perceptrón multicapa que realiza:

- Clasificación (*softmax*)
- Regresión (lineal):
  - *Bounding Box*



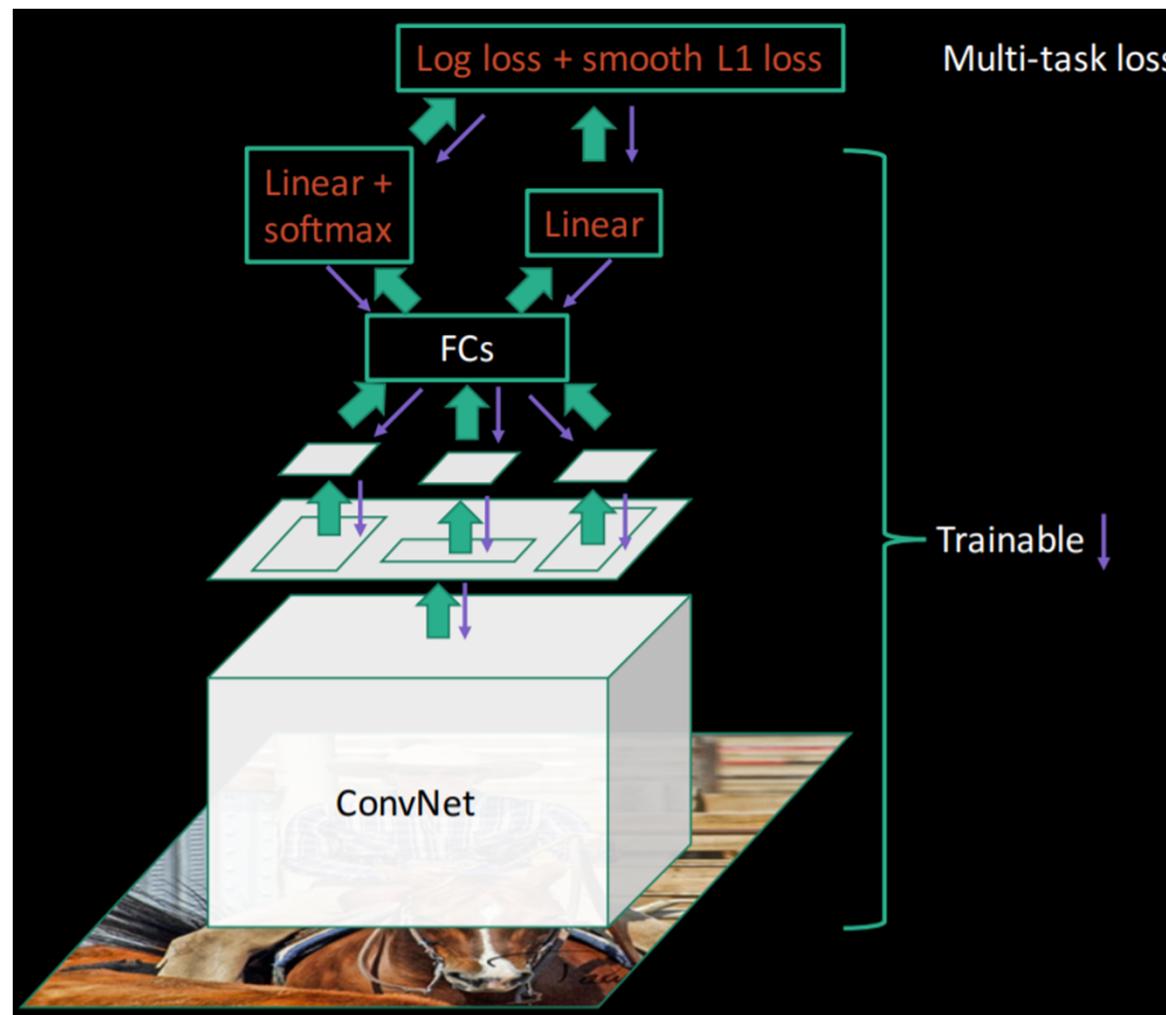
# DEEP LEARNING

- *Fast RCNN:*



# DEEP LEARNING

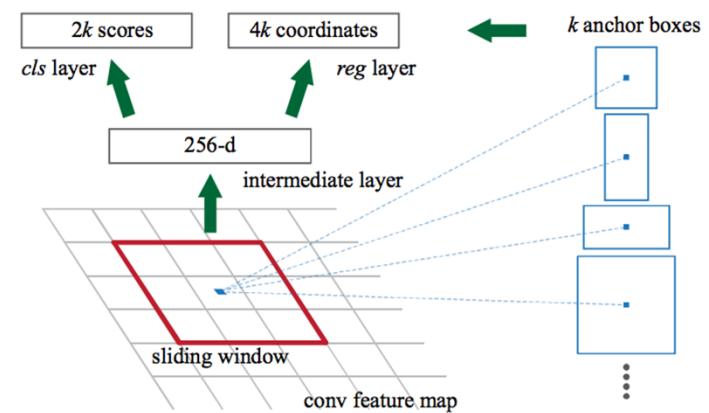
- *Fast RCNN:*



# DEEP LEARNING

- *Faster RCNN:*

- En lugar de usar *Selective Search*, se utiliza una *Region Proposal Network (RPN)* para seleccionar regiones de interés
  - Utiliza una ventana deslizante y, de cada ventana, extrae  $k$  cajas de distintos tamaños y formas
  - De cada caja, predice la probabilidad de que contenga un objeto y el bounding box

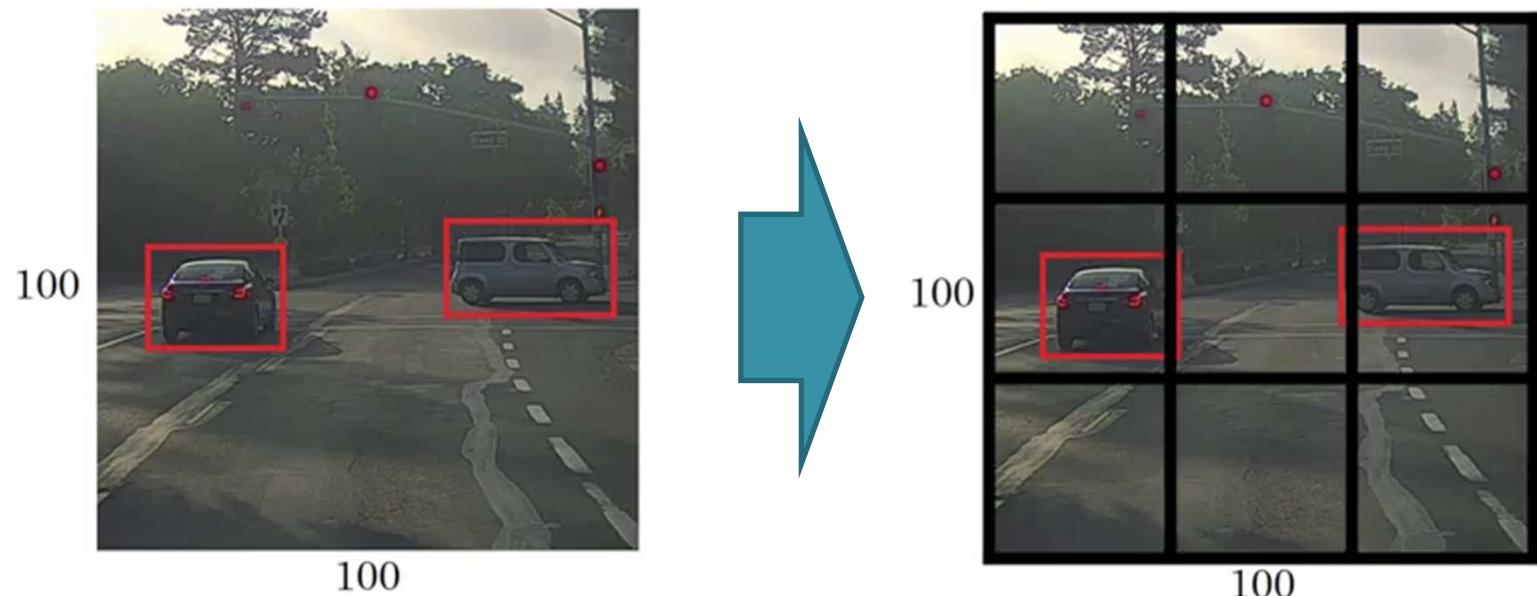


# DEEP LEARNING

- **YOLO (You Only Look Once)**

1. Se toma la imagen y se divide en una rejilla

- En este ejemplo, 3x3, pero suelen ser mucho más pequeñas



2. Se utiliza un algoritmo (Selective Search) para tomar las regiones de interés

# DEEP LEARNING

- **YOLO (You Only Look Once)**

- 2. De cada celda, se crean etiquetas:

- Probabilidad de que haya un objeto (1 salida: 1/0)
    - Bounding Box (4 salidas)
    - Etiquetas de clase (1 salida por clase)
  - Por ejemplo, si se quiere detectar peatón/coche/moto, se necesitan 8 salidas:



$y =$	0
	?
	?
	?
	?
	?
	?
	?



$y =$	1
	$bx$
	$by$
	$bh$
	$bw$
	0
	1
	0

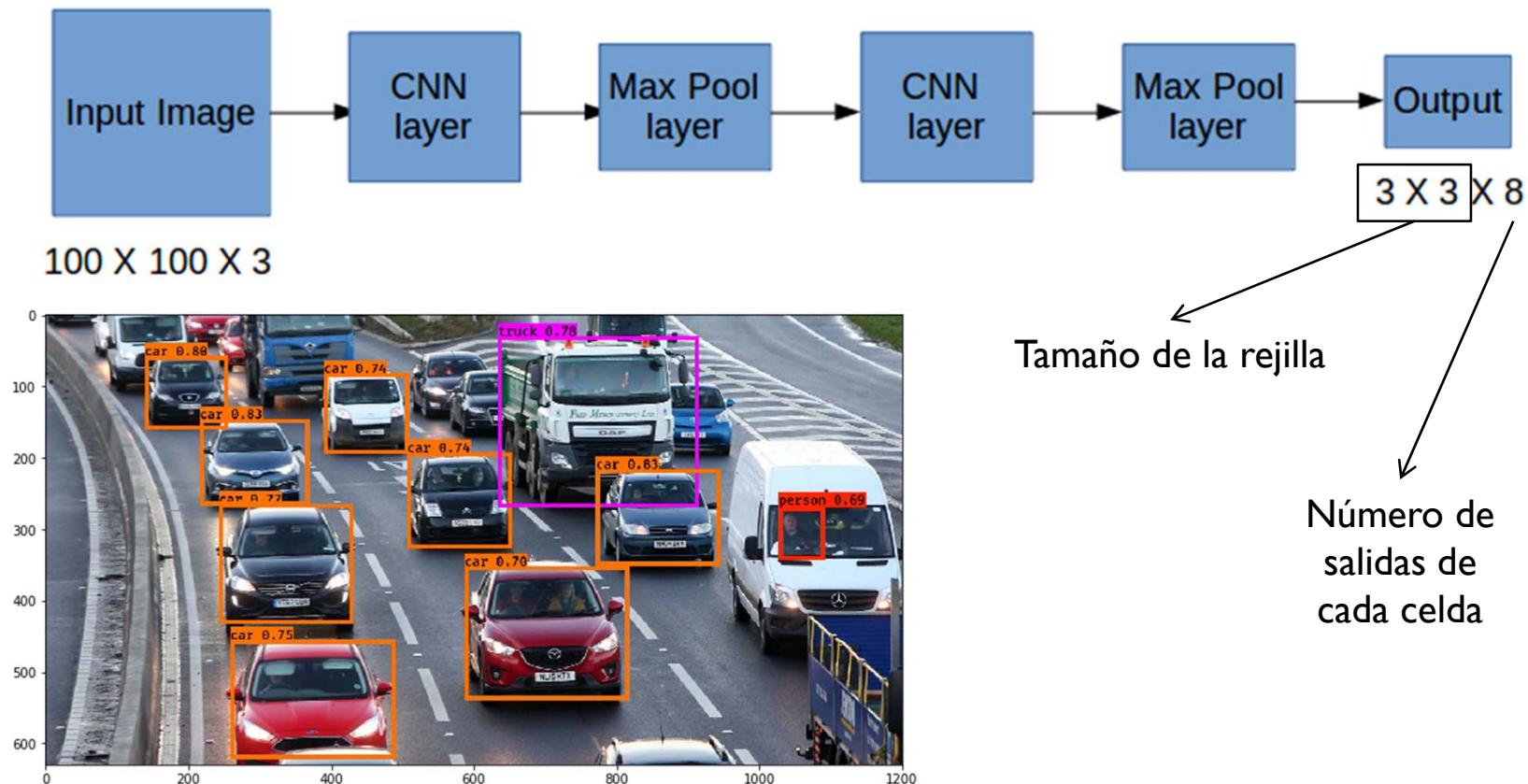
- Un objeto pertenece a la celda donde está su punto central
    - Si hay más de un objeto en una celda, dividir en una rejilla más pequeña (con más celdas)

# DEEP LEARNING

- YOLO (*You Only Look Once*)

- 3. Entrenar el modelo:

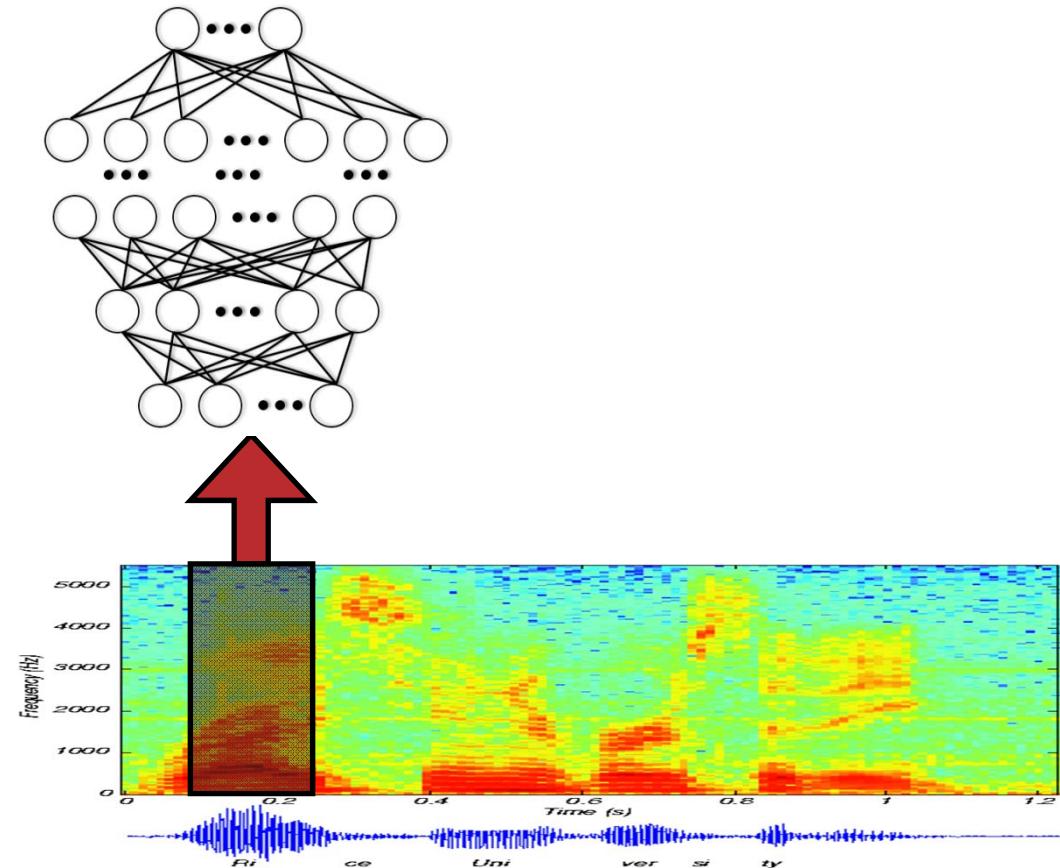
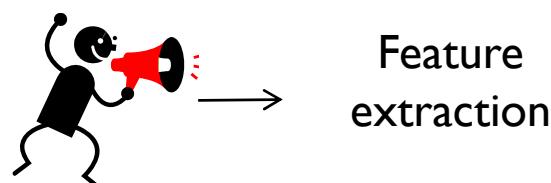
- Ejemplo anterior:



# DEEP LEARNING

- Otras aplicaciones:
  - Habla:

Phone  
probabilities





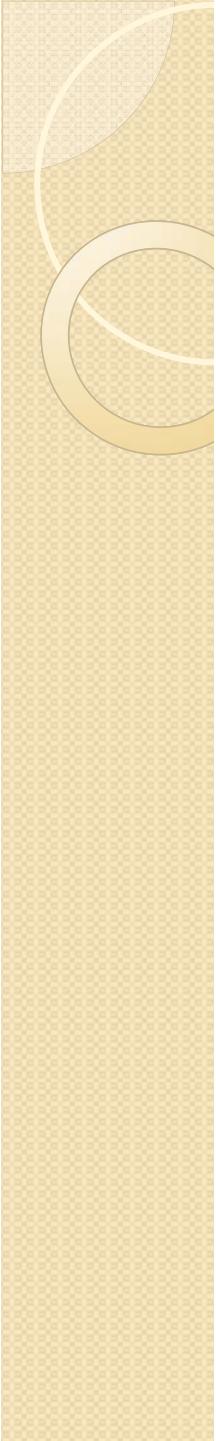
# DEEP LEARNING

- Cuestiones abiertas:
  - ¿Es así como funciona el cerebro?
  - Hace mucho que se conocen las redes de neuronas, ¿por qué este resurgimiento ahora?
    - ¿Capacidad computacional?
    - ¿Más datos disponibles?
    - ¿Conexión con neurociencia?
  - ¿Se puede emular el cerebro computacionalmente?
    - $\sim 10^{11}$  neuronas,  $\sim 10^{15}$  conexiones
      - Mayor RNA:  $\sim 10^4$  neuronas,  $\sim 10^8$  conexiones
      - Muchas conexiones van hacia atrás
      - La comprensión del cerebro está muy lejos de ser completa



# DEEP LEARNING

- *Deep Learning* y *Feature Learning* hoy en día:
  - Reconocimiento de voz
    - Ha sido el tema de investigación más “caliente” en los últimos años
    - Técnicas anteriores que llevaban muchos años han sido mejoradas
    - Microsoft y Google utilizan en sus productos sistemas basados en *Deep Learning*
    - Microsoft, Google, IBM, Nuance, AT&T y la mayoría de entidades académicas e industriales tienen proyectos de *Deep Learning*



# DEEP LEARNING

- *Deep Learning y Feature Learning* hoy en día:
  - Visión Artificial
    - La ingeniería de características siempre ha sido la línea central de investigación de una gran parte de la comunidad de investigadores en VA
    - Esto ha creado cierta resistencia al *Feature Learning*
    - Las redes convolucionales constituyen una técnica de segmentación de imágenes y de VA de grandes resultados
  - Procesamiento de lenguaje natural



# DEEP LEARNING

- *Deep Learning y Feature Learning hoy en día:*
  - En muchos campos, Deep Learning ha supuesto una revolución
    - Métodos usados en sistemas comerciales
      - Reconocimiento de caras y gente
      - Reconocimiento de objetos de baja resolución
        - Señales de tráfico, números de casas, etc.
      - Reconocimiento de voz
      - etc.