Deep learning: Redes convolucionales

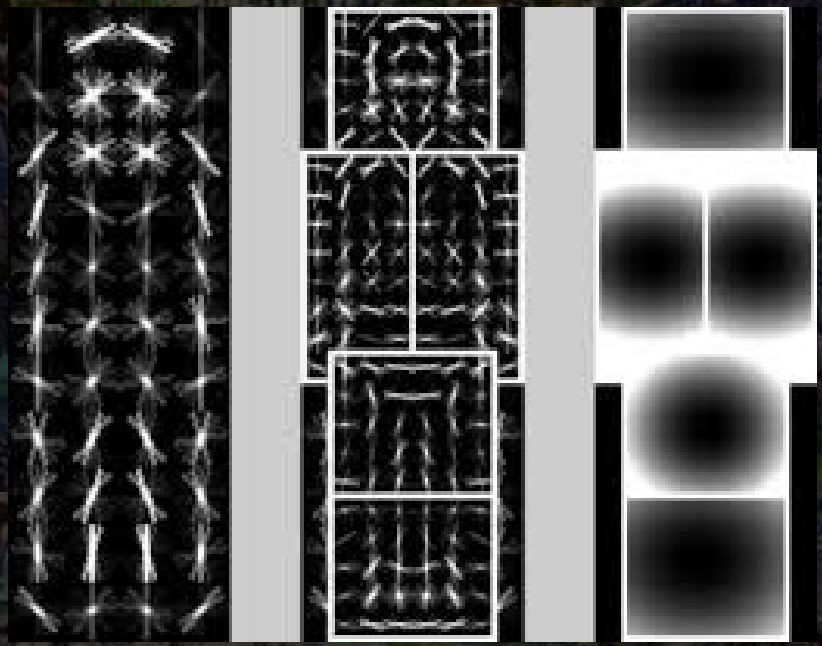
Patricio Loncomilla

1 Introducción

- Los algoritmos basados en Deep Learning (como las redes feed forward) tienen en teoría la capacidad de aproximar cualquier función
- Esa propiedad teórica indica que existe una red óptima, pero no implica que el proceso de aprendizaje sea capaz de alcanzar la configuración óptima
- En la práctica las redes no funcionaban bien y los algoritmos basados en Deep Learning fueron abandonados

1. Introducción

- Algoritmos usando características diseñadas a mano dominaron por mucho tiempo el campo de la inteligencia computacional
- Performance limitada al escoger modelo
- Requieren conocimiento experto

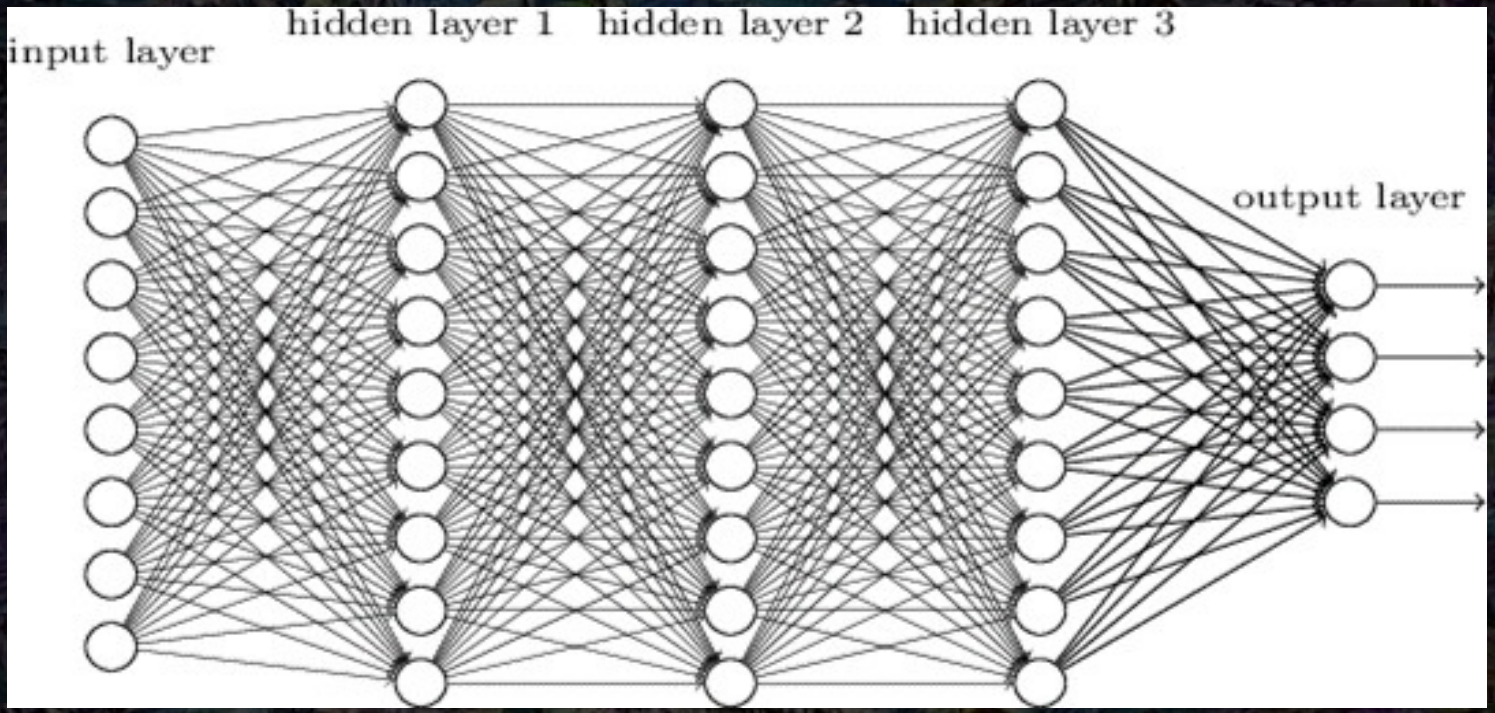


1. Introducción

- En los últimos años, las redes deep han revivido
- Mejoras en los algoritmos y la GPU
- Mucha mejor performance que los algoritmos clásicos

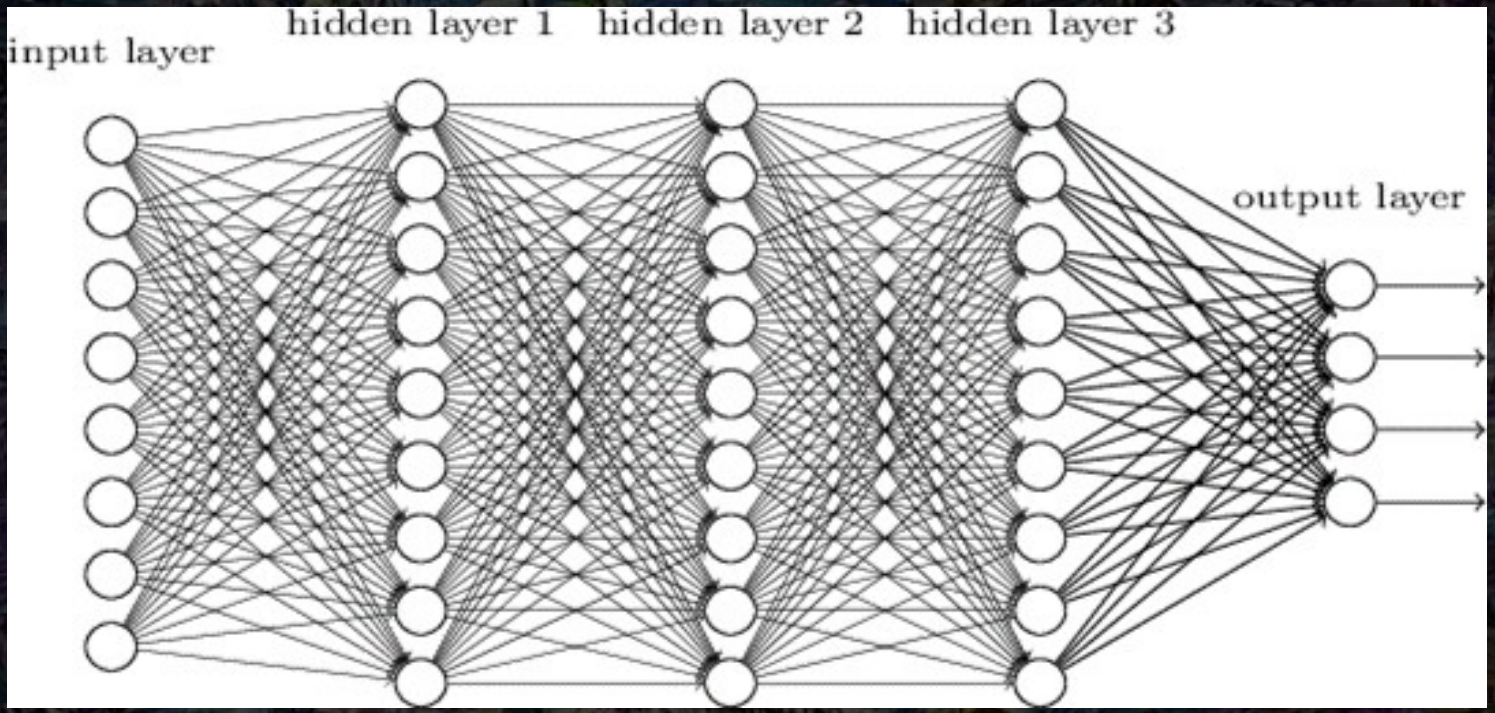
2 Redes deep feedforward

- Son redes que son capaces de aprender relaciones entrada-salida a partir de una gran cantidad de ejemplos



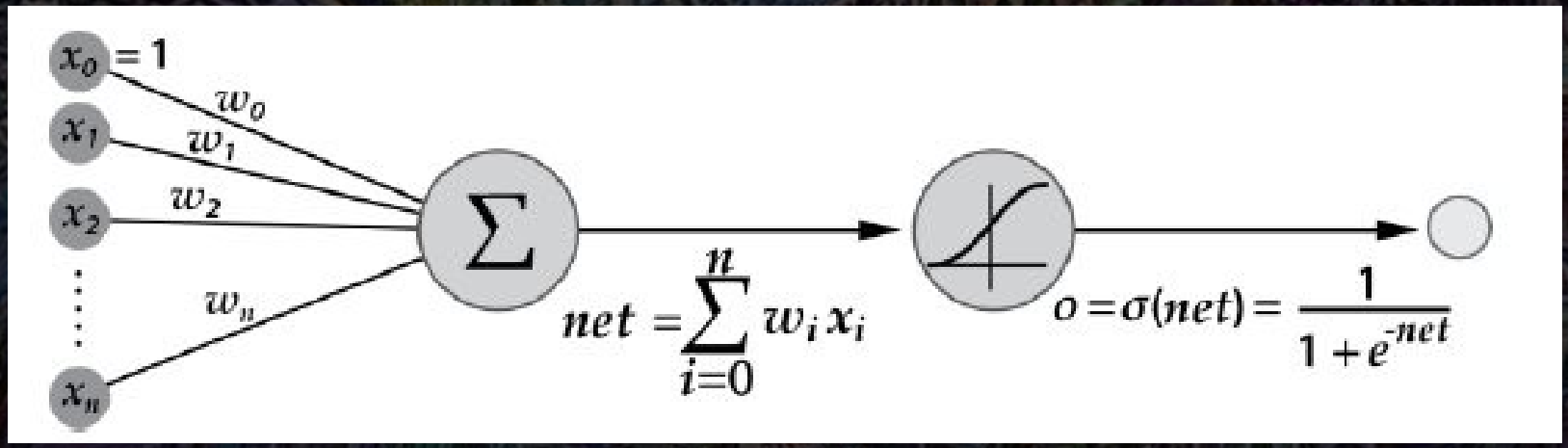
2 Redes deep feedforward

- Están formadas por muchas neuronas simples interconectadas entre sí
- Deep => muchas capas



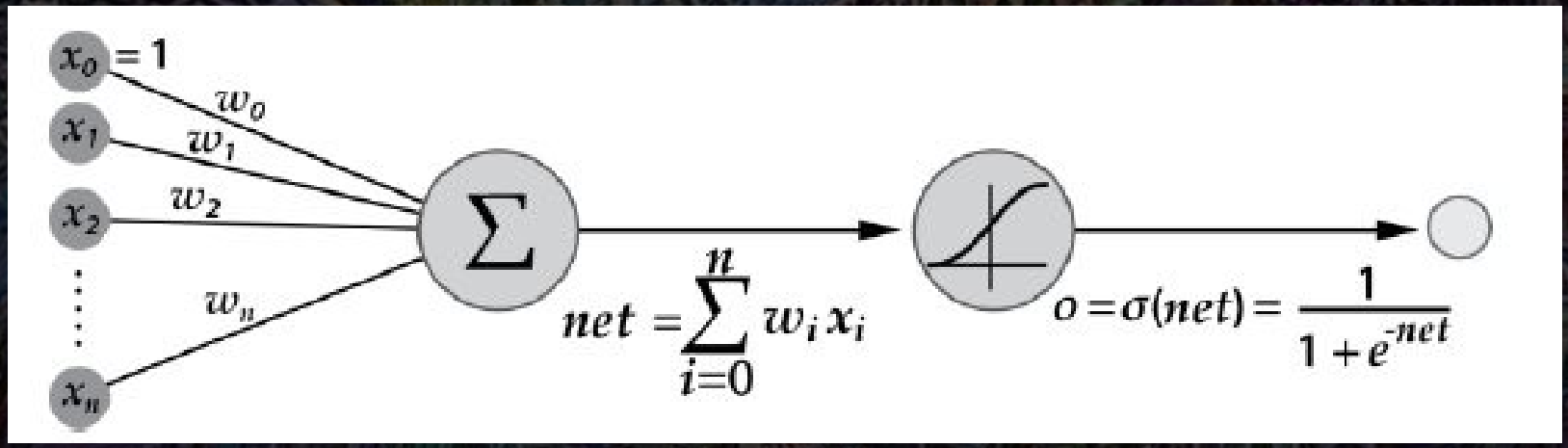
2.1 Neuronas

- La neurona consiste de un modelo lineal seguido por una no-linealidad
- Cada neurona tiene varias entradas y una salida



2.1 Neuronas

- Cada entrada es multiplicada por un peso sináptico
- Neurona determinada por pesos y no-linealidad



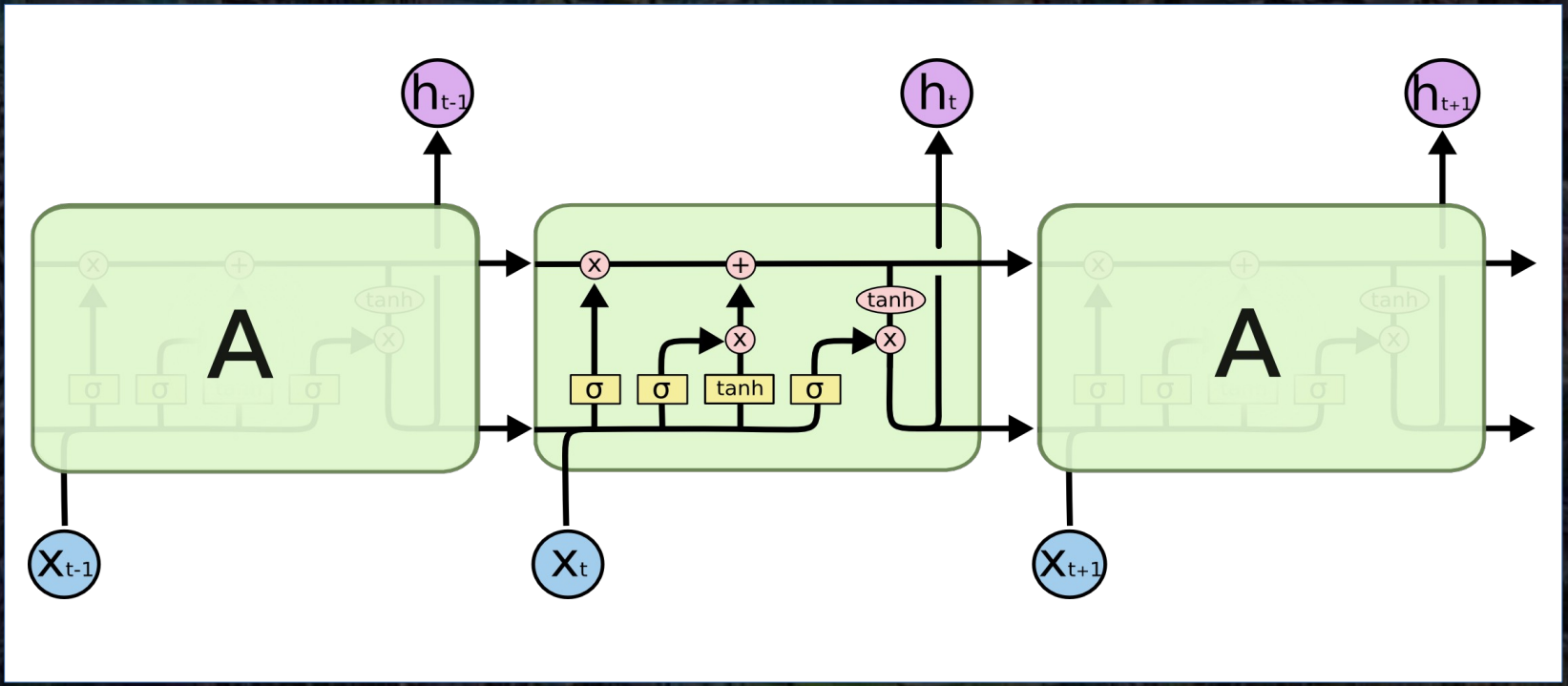
2.1 Neuronas

- No linealidades comunes (muchas)
- Sigmoidal logística

- RELU

2.1 Neuronas

- También hay neuronas con modelos más complejos, ej:
- Long short-term memory (memory cells)



http://deeplearning.cs.cmu.edu/pdfs/Hochreiter97_lstm.pdf

2.2 División de los datos

- El sistema se debe entrenar y probar con un conjunto de datos
- Cada dato corresponde a una entrada y una salida
- Se debe dividir el conjunto de datos en tres:
  - Conjunto de entrenamiento
  - Conjunto de validación
  - Conjunto de prueba

2.2 División de los datos

- Conjunto de entrenamiento:
  - Permite obtener los pesos de una red dada
  - Ajuste de parámetros
- Conjunto de validación:
  - Permite comparar el desempeño de distintas arquitecturas de red
  - Ajuste de hiperparámetros
- Conjunto de prueba
  - Permite estimar el desempeño de la red en datos futuros => El mejor modelo se debe probar acá

2.3 Inicialización de pesos

- Antes de comenzar el entrenamiento, es necesario elegir los pesos iniciales
- Los pesos para cada neurona dependen del número de entradas
- Se usa una distribución uniforme

$$W \sim U\left[-\sqrt{2/n_i}, \sqrt{2/n_i}\right]$$

2.4 Entrenamiento

- El entrenamiento consiste en minimizar una función de pérdida

$$min\, L(x, y, w)$$

- La función de pérdida puede consistir por ejemplo en el error de aproximación

$$L(x, y, w) = E[(y - f(x, w))^2]$$

- La función de pérdida depende del problema

2.4 Entrenamiento

- En el caso de las redes feed-forward, se suele usar un método del gradiente estocástico para resolver el problema

$$w_t = w_{t-1} - \lambda * \frac{dL(x, y, w_{t-1})}{dw}$$

- El hiperparámetro λ se denomina tasa de aprendizaje
- Se usan pocos ejemplos en cada iteración

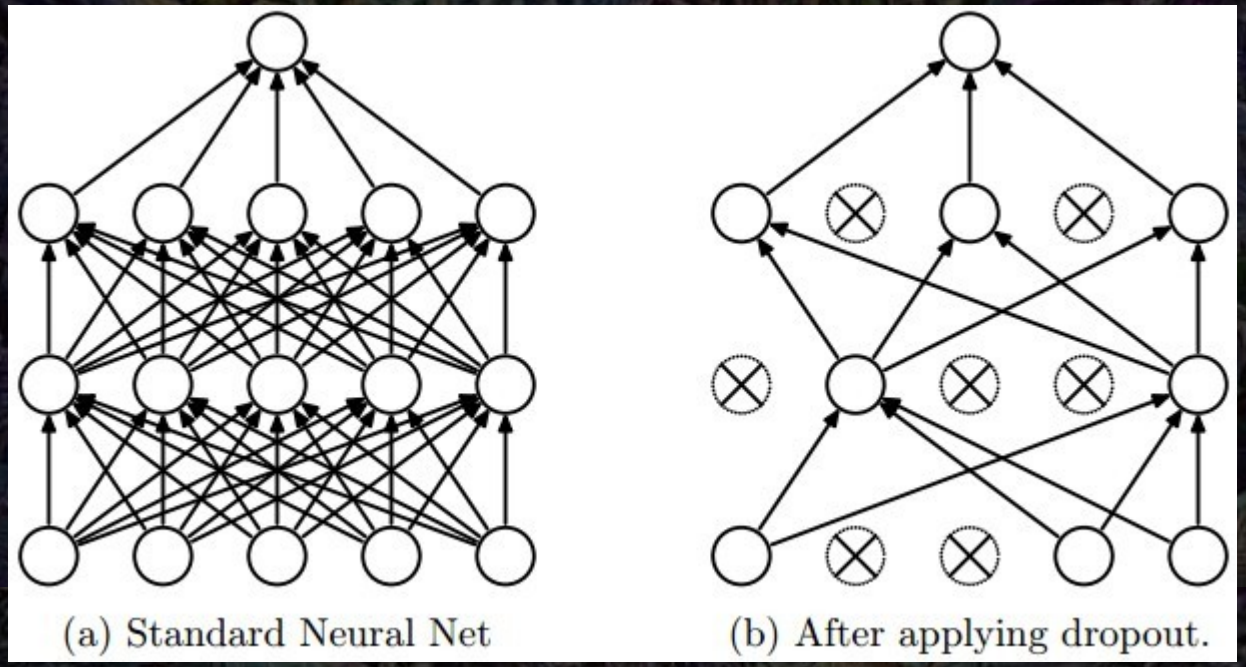
2.4 Entrenamiento

- Regularización:
- Consiste en agregar términos extra a la función de pérdida
- Permite mejorar la capacidad de generalización

$$L_{reg}(x, y, w) = L(x, y, w) + \alpha * |w|^2$$

2.4 Entrenamiento

- Dropout: Consiste en desconectar un porcentaje de las neuronas en cada iteración del entrenamiento
- Mejora la generalización de la red



(a) Standard Neural Net (b) After applying dropout.

2.4 Entrenamiento

- Resumen
  - Inicializar pesos
  - Método del gradiente estocástico
  - Regularización
  - Dropout
- Junto a ReLU, GPUs y bases de datos grandes permiten buen rendimiento de la red

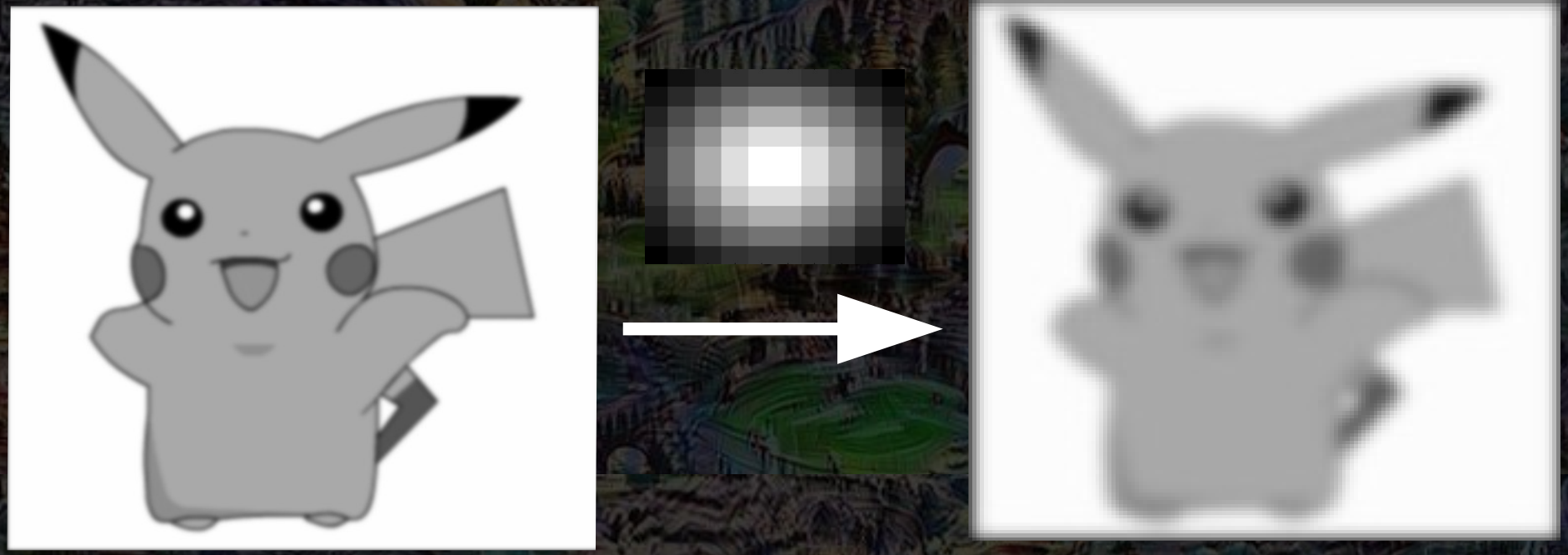
3 Redes convolucionales

- Son redes que se usan para procesar imágenes
- Pueden aprender relaciones entrada-salida, donde la entrada es una imagen
- Estan basadas en operaciones de convolución
- Tareas comunes:
  - Detección/categorización de objetos
  - Clasificación de escenas
  - Clasificación de imágenes en general

3 Redes convolucionales

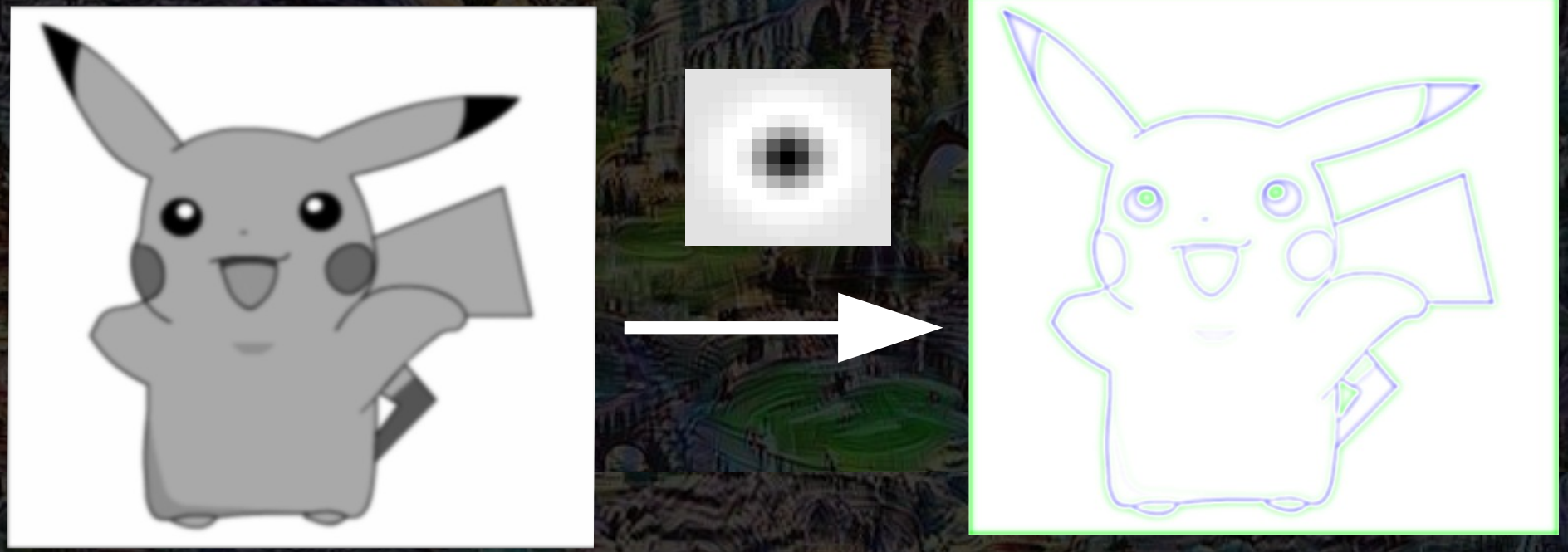
- Convolución: Consiste en filtrar una imagen usando una máscara
- Diferentes máscaras producen distintos resultados

Gaussiana



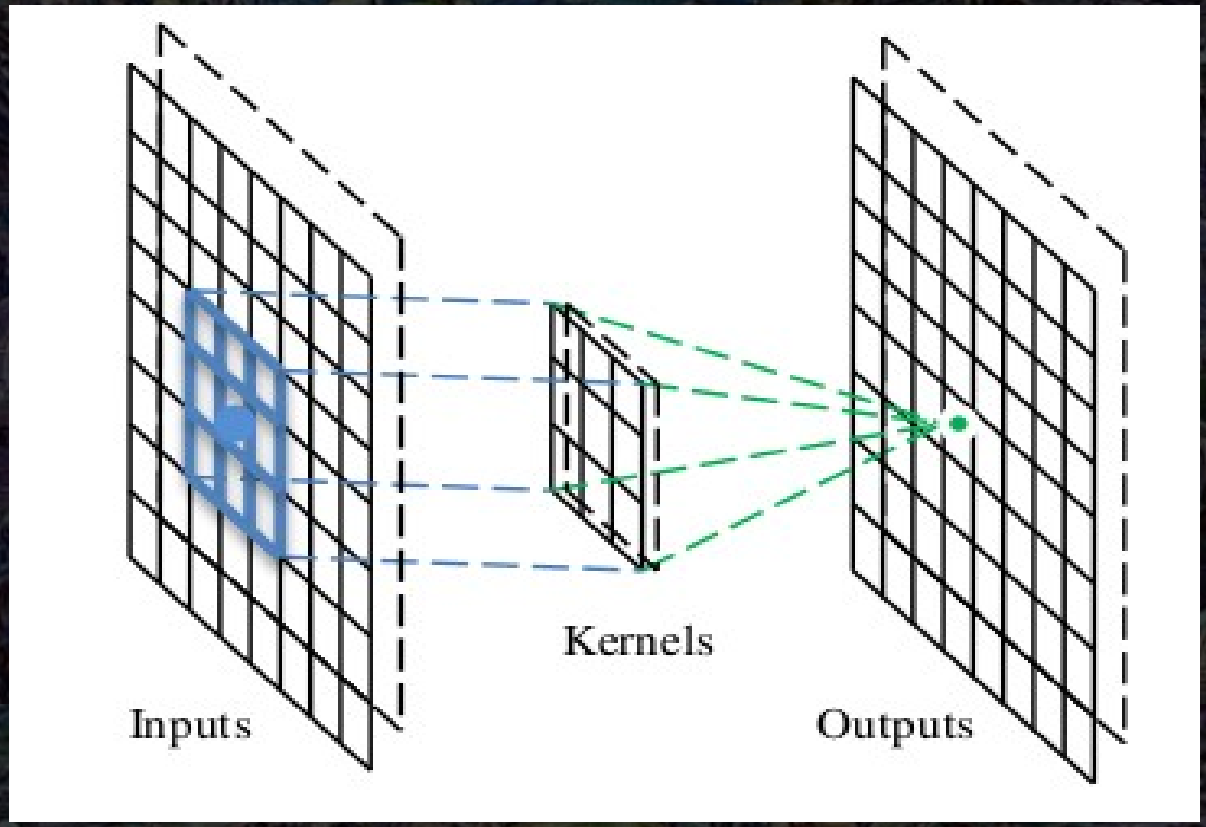
3 Redes convolucionales

- Convolución: Consiste en filtrar una imagen usando una máscara
- Diferentes máscaras producen distintos resultados



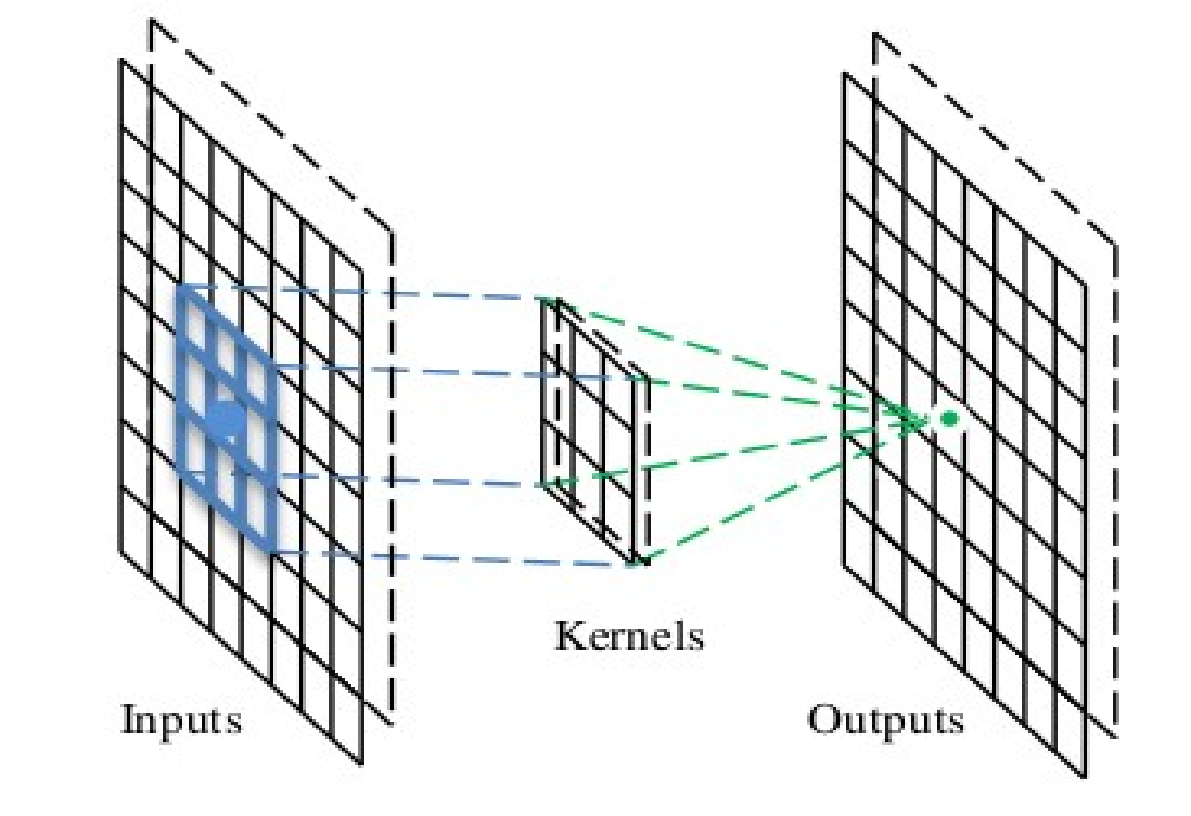
3 Redes convolucionales

- En la convolución, cada pixel de salida es una combinación lineal de los pixeles de entrada
  - Ventana deslizante



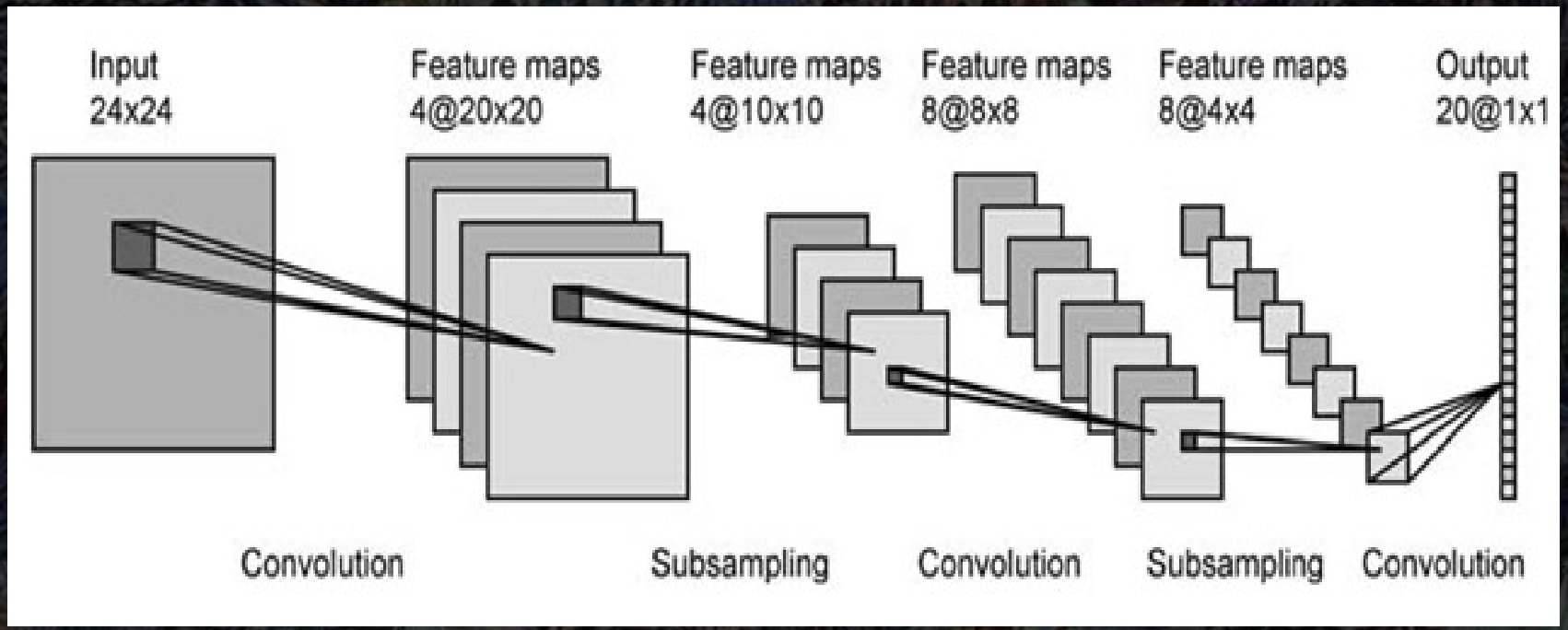
3 Redes convolucionales

- Las máscaras representan la conectividad entre las capas sucesivas
- Pesos = valores en las máscaras



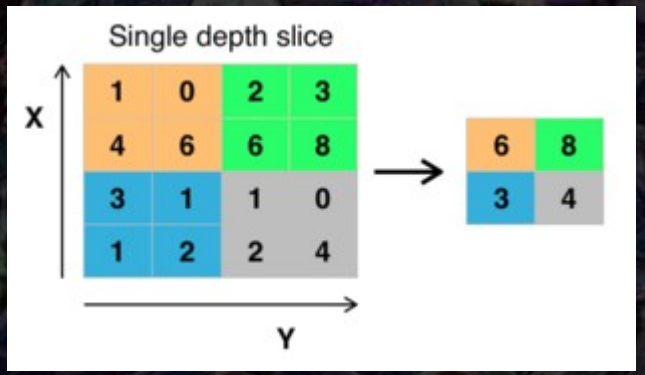
4 Arquitecturas

- Cada capa es un volumen de neuronas en 3D
- Dimensiones:
  - Alto, ancho, profundidad de la capa



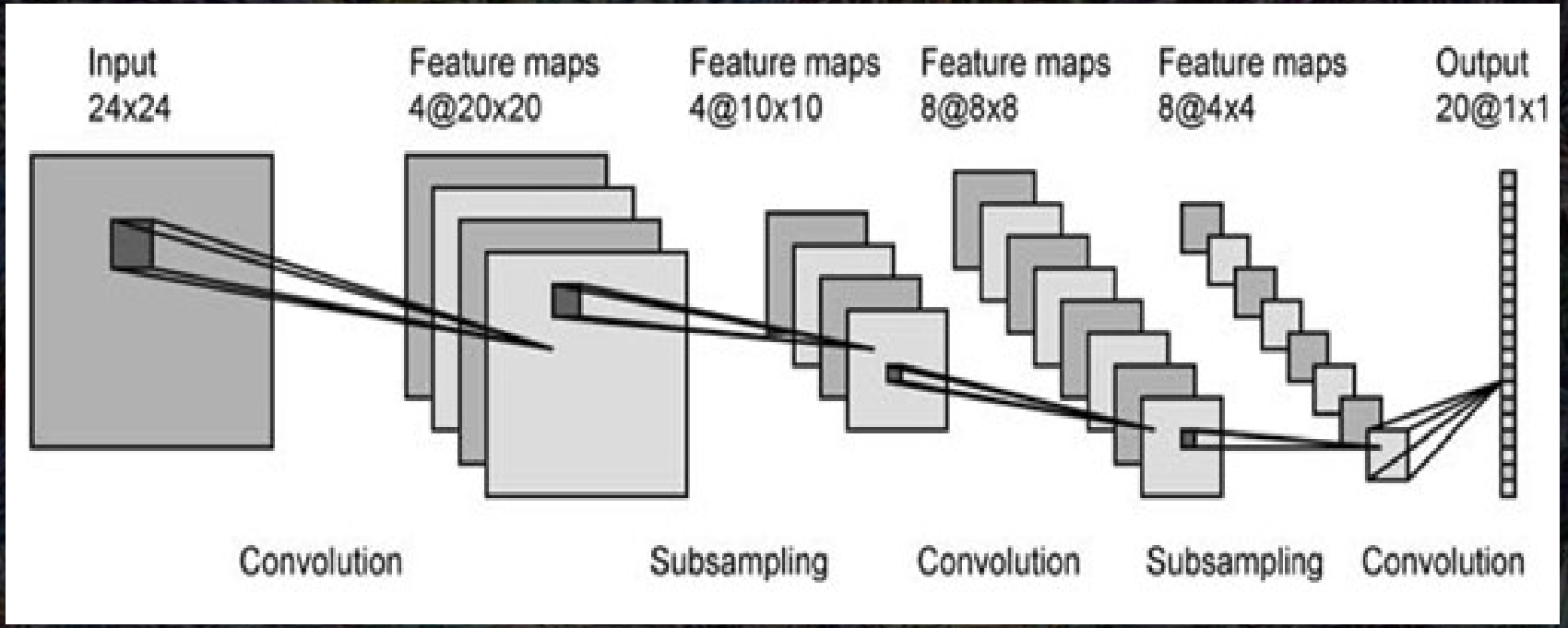
3 Redes convolucionales

- Las redes se forman usando tres tipos de capas:
  - Capas convolucionales
    - Requieren el uso de máscaras
    - Filtrado + ReLU
  - Capas de pooling
    - La salida es el máximo de la entrada en una ventana
    - Se puede usar para submuestrear (stride)
  - Capas totalmente conectadas
    - Se aplican al final, se pierde la información espacial



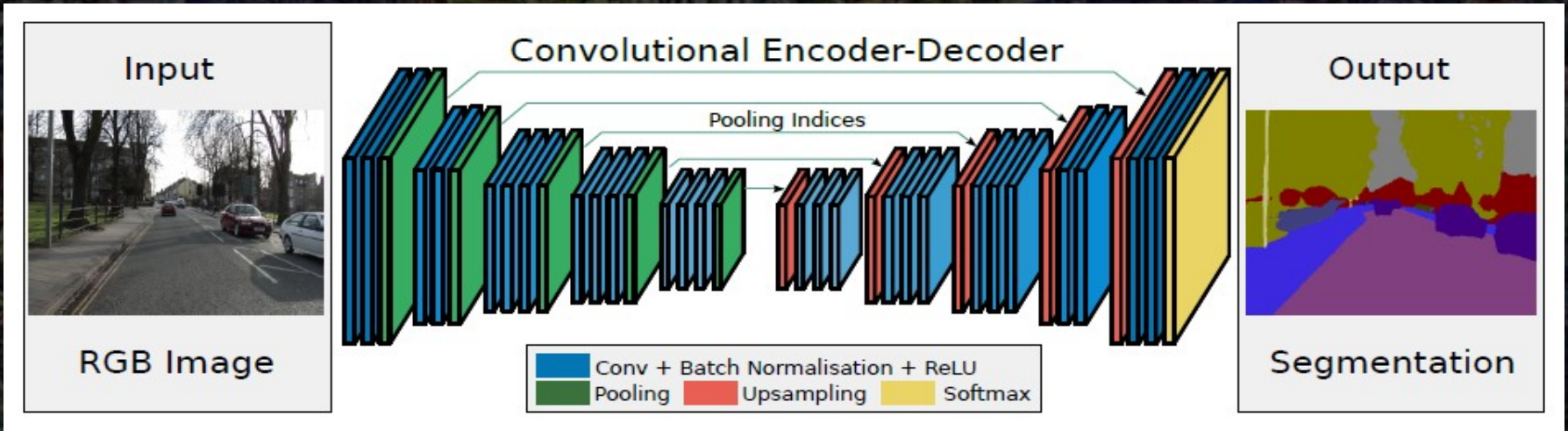
4 Arquitecturas

- Hay dos arquitecturas básicas de CNN:
  - CNN:
    - Entrega una salida para toda la imagen
    - Salida fully connected (todas conectadas con todas)



4 Arquitecturas

- Hay dos arquitecturas básicas de CNN:
  - Fully convolutional networks:
    - Tienen un encoder y un decoder
    - Comprimen la información
    - Entregan una salida por píxel



4 Arquitecturas CNN

- AlexNet
- Entrenado con ImageNet ILSVRC-2012
  - 1.2 millones de imágenes
  - 1000 clases
  - Entrenado usando 2 GPUs durante una semana

http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

4 Arquitecturas CNN

- AlexNet (TorontoNet)

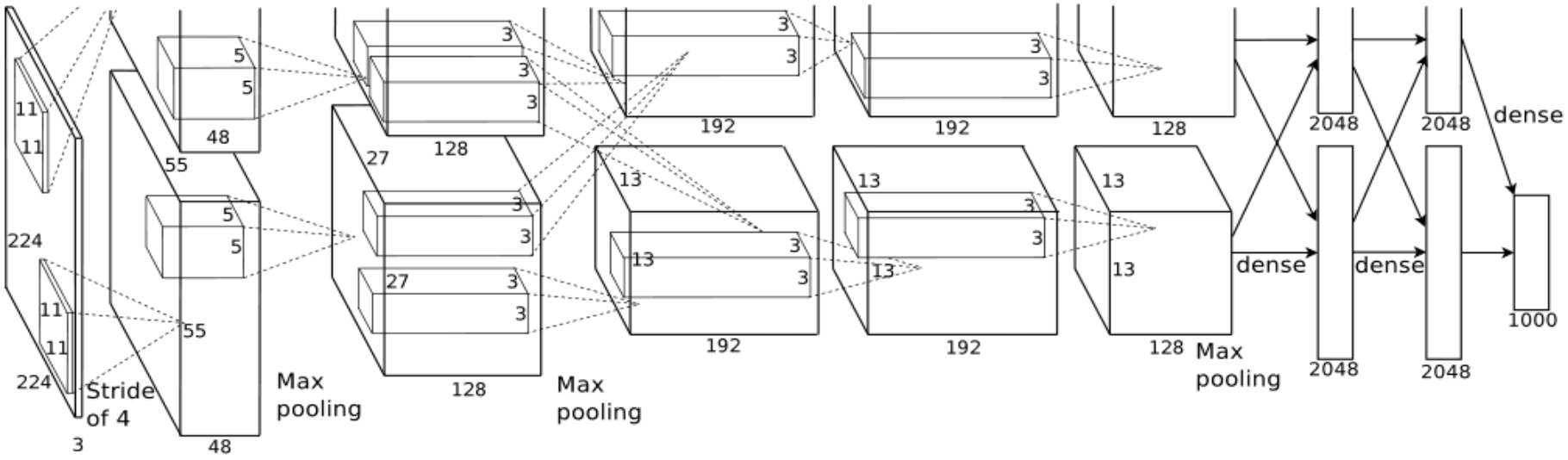
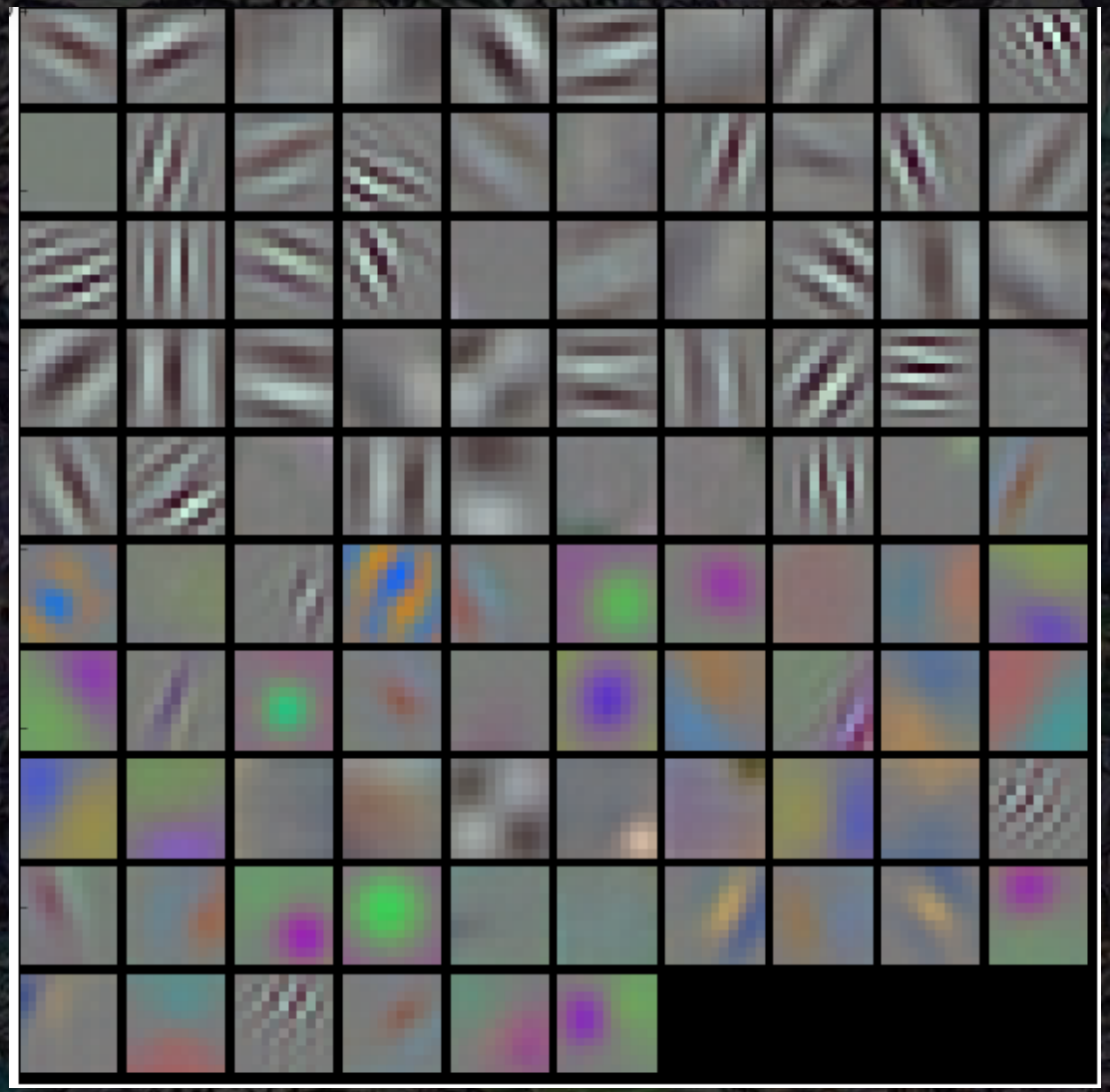


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

4 Arquitecturas CNN

- Ejemplo de arquitectura: AlexNet



4 Arquitecturas CNN

- Zeiler (Clarifai)
  - 5 conv + 3 fc

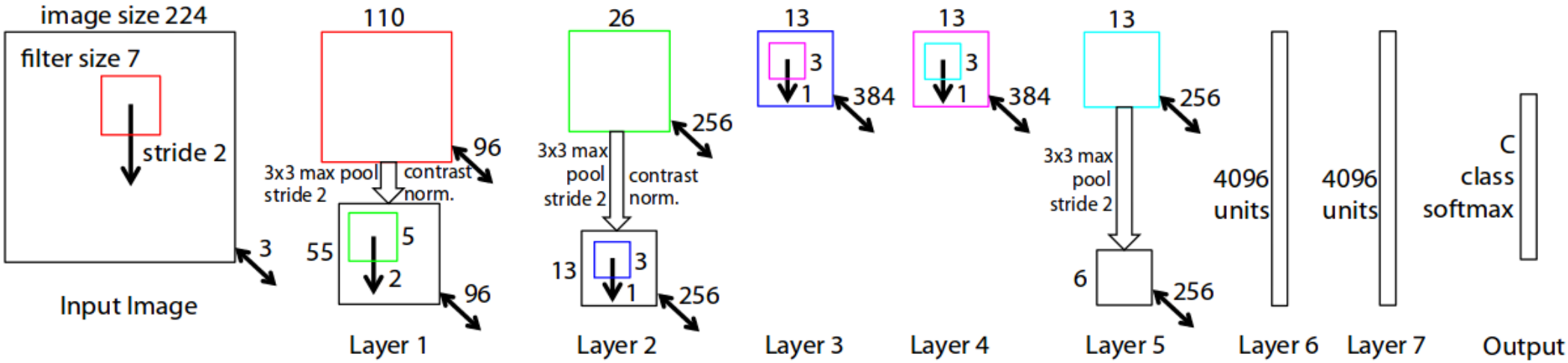


Figure 3. Architecture of our 8 layer convnet model. A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through a rectified linear function (not shown), (ii) pooled (max within 3x3 regions, using stride 2) and (iii) contrast normalized across feature maps to give 96 different 55 by 55 element feature maps. Similar operations are repeated in layers 2,3,4,5. The last two layers are fully connected, taking features from the top convolutional layer as input in vector form ($6 \cdot 6 \cdot 256 = 9216$ dimensions). The final layer is a C-way softmax function, C being the number of classes. All filters and feature maps are square in shape.

http://www.matthewzeiler.com/pubs/arxive2013/arxive2013.pdf

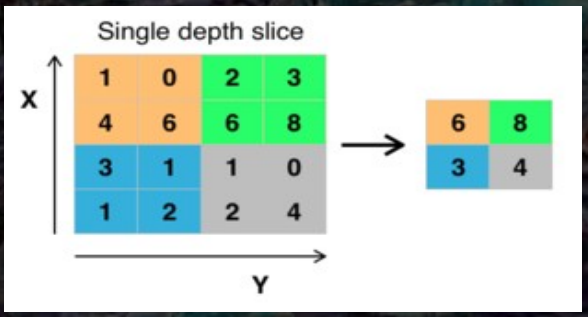
4 Arquitecturas CNN

- VGG16
  - Tiene más capas

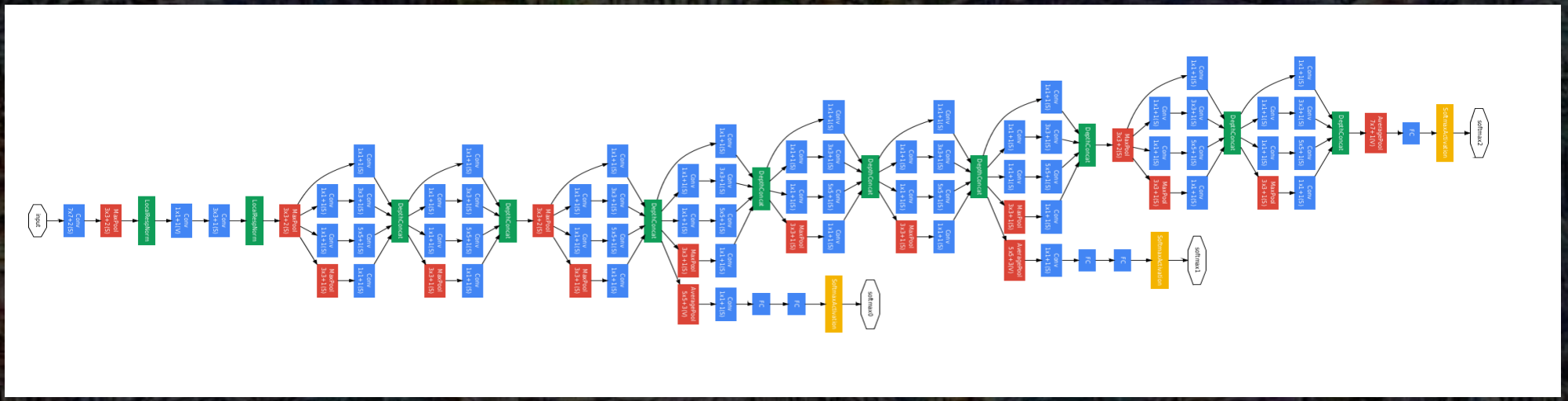
Model	VOC-2007 (mean AP, %)	VOC-2012 (mean AP, %)	Caltech-101 (mean class recall, %)	Caltech-256 (mean class recall, %)
16-layer	89.3	89.0	91.8±1.0	85.0±0.2
19-layer	89.3	89.0	92.3±0.5	85.1±0.3
model fusion	89.7	89.3	92.7±0.5	86.2±0.3

http://www.robots.ox.ac.uk/~vgg/research/very_deep/

4 Arquitecturas CNN

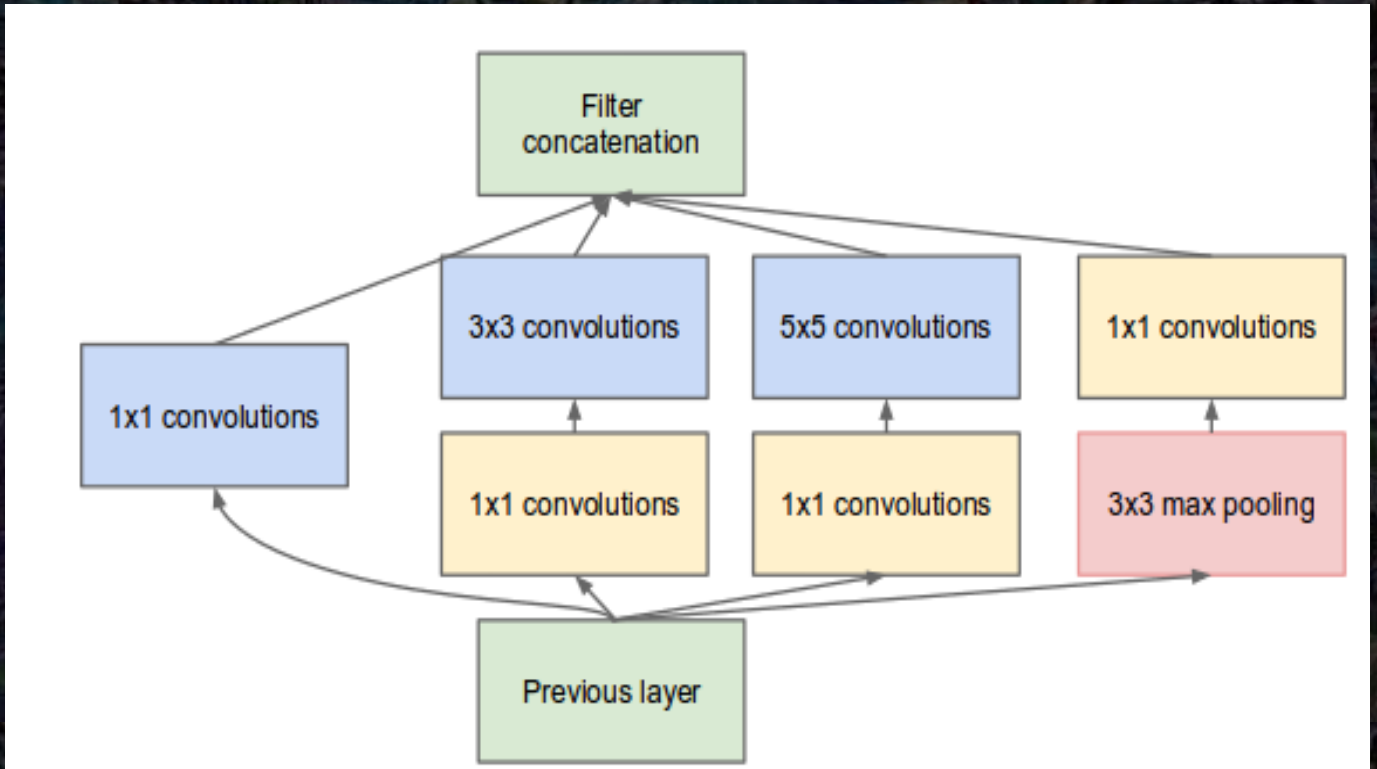


- GoogLeNet
  - 22 layers
  - Neurona con modelo más complejo NIN
    - Varios cálculos en paralelo
    - Pooling sin reducir la imagen



4 Arquitecturas CNN

- GoogLeNet
  - Se usan varias resoluciones



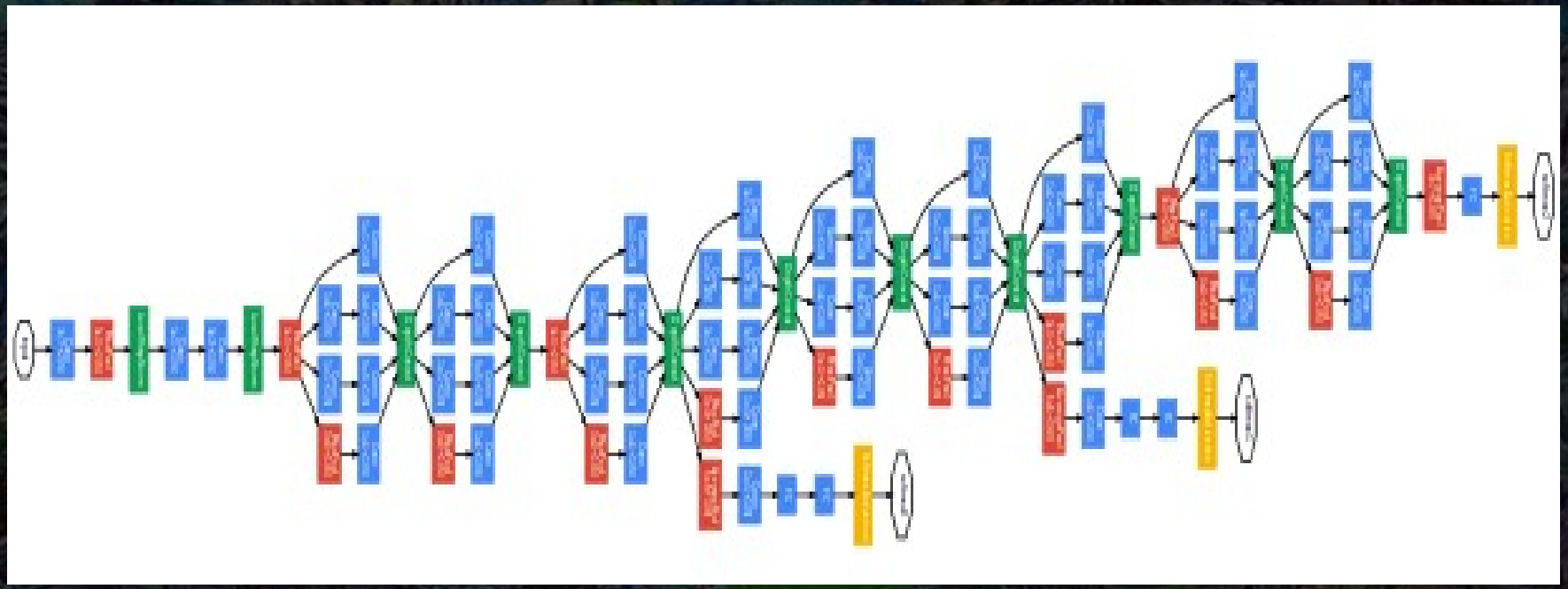
(b) Inception module with dimensionality reduction

4 Arquitecturas CNN

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

4 Arquitecturas CNN

- GoogLeNet
- La arquitectura final no es simplemente una sucesión de capas convolucionales

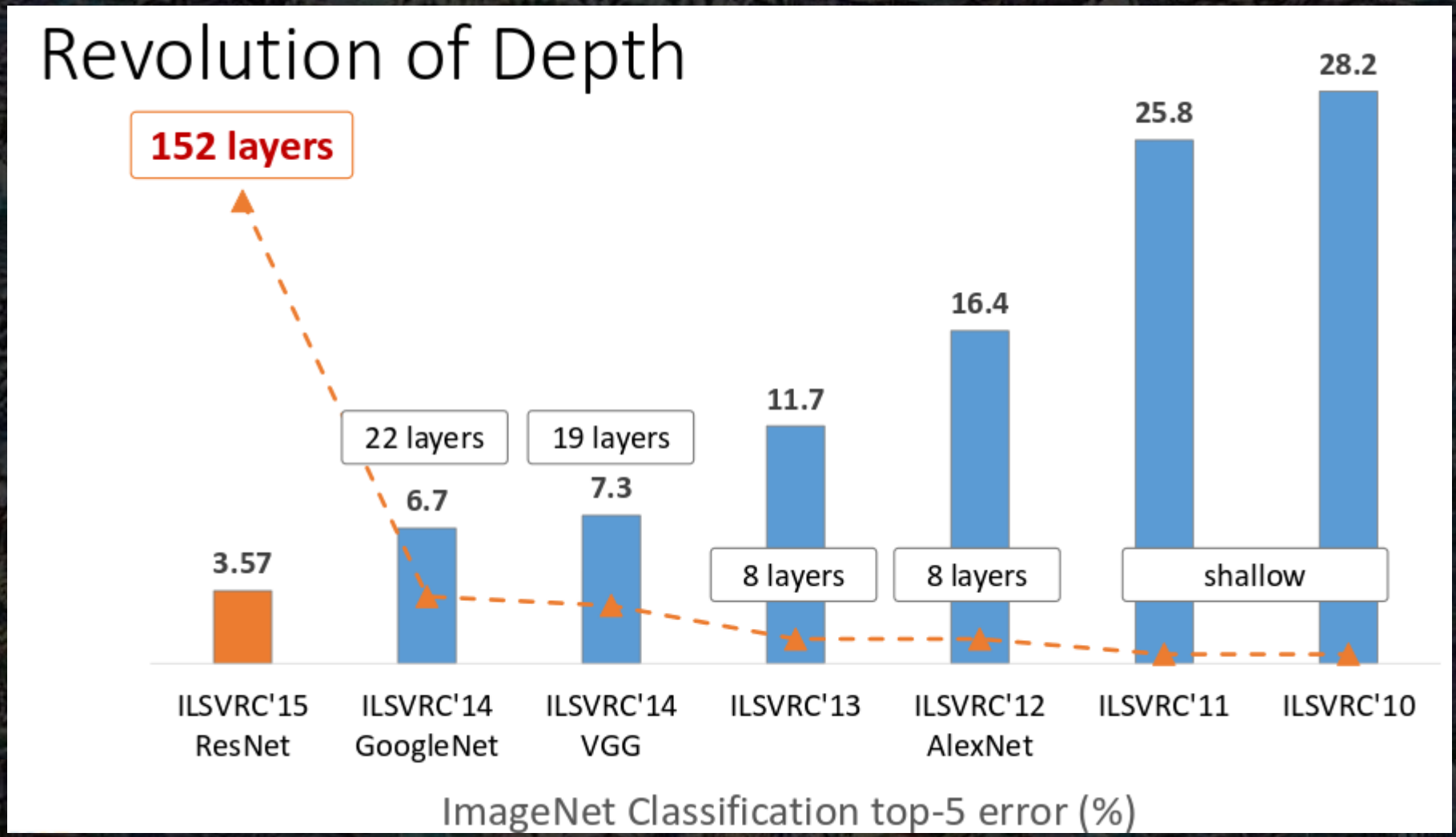


4 Arquitecturas CNN

- ResNet (MSRA 2015)
  - Estado de arte en CNN
  - “Deep residual learning”
  - 152 layers en ICCV 2015
  - 1001 layers actualmente

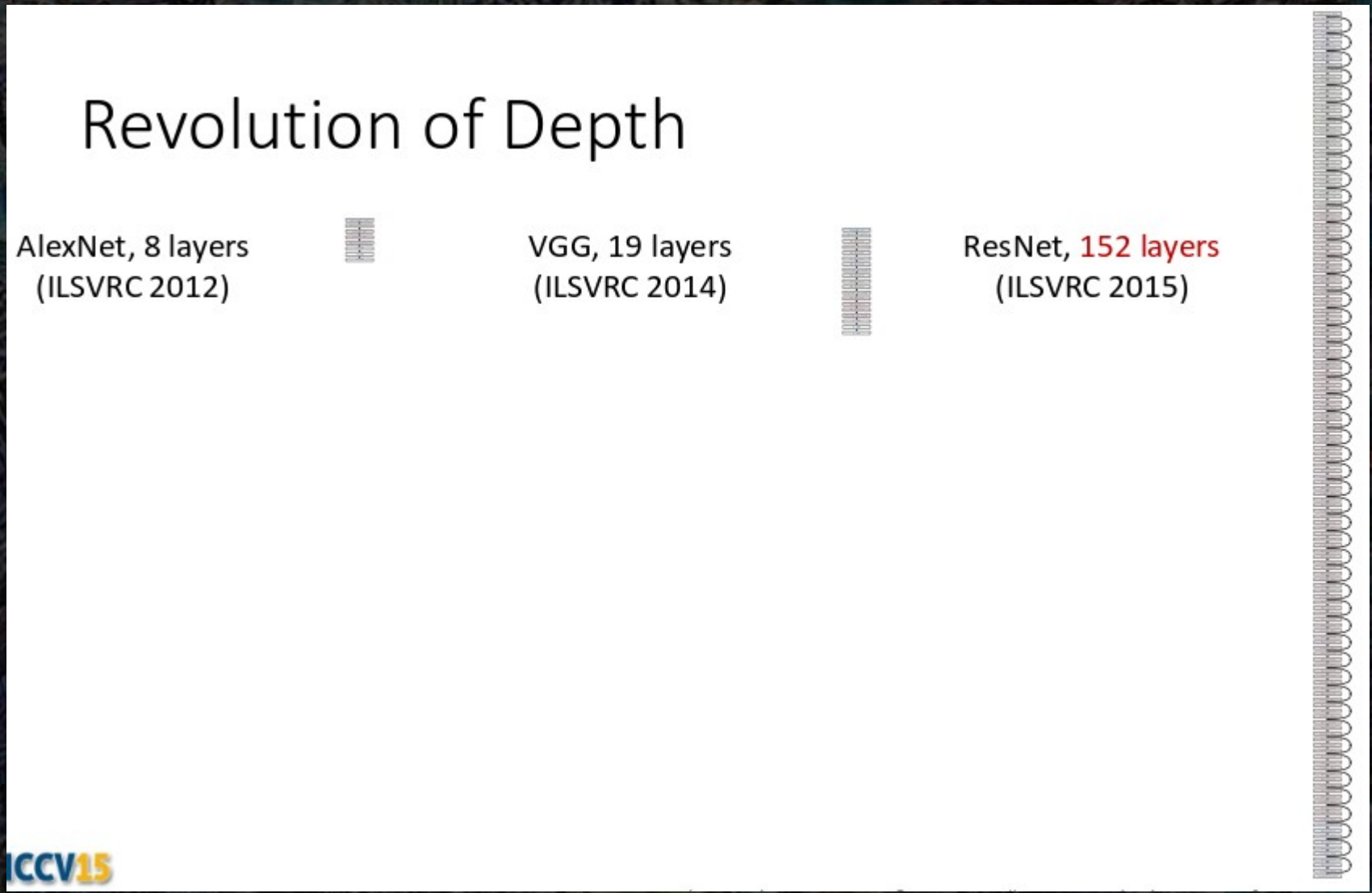
4 Arquitecturas CNN

- ResNet



4 Arquitecturas CNN

- ResNet

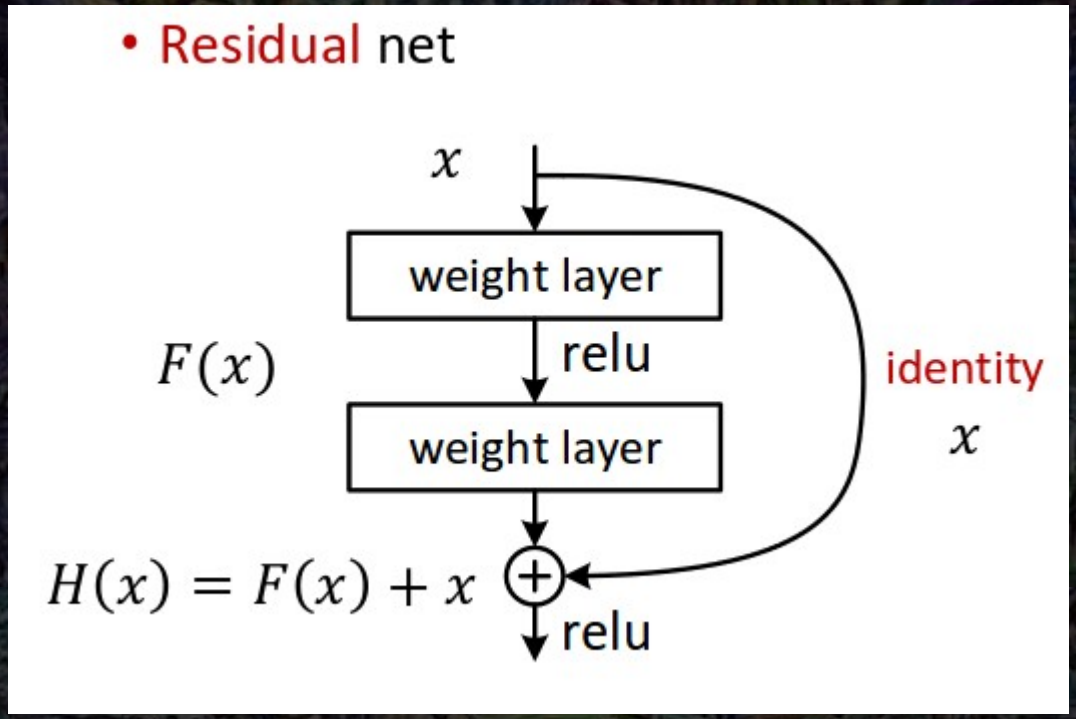


4 Arquitecturas CNN

- ResNet
- Al agregar más de 30 capas, el error comienza a aumentar en vez de disminuir
- Esta es la causa de que haya que usar modelos complejos como GoogleNet
- Encontraron una solución a ese problema

4 Arquitecturas CNN

- ResNet
- La idea es que las capas extras reduzcan efectivamente el error
- Se trabaja con residuales



4 Arquitecturas CNN

- ResNet
- A diferencia de GoogLeNet, el diseño es simple
- Sólo se usan capas convolucionales y max poolings
- El uso de residuales permite usar muchas capas (actualmente 1001 capas)

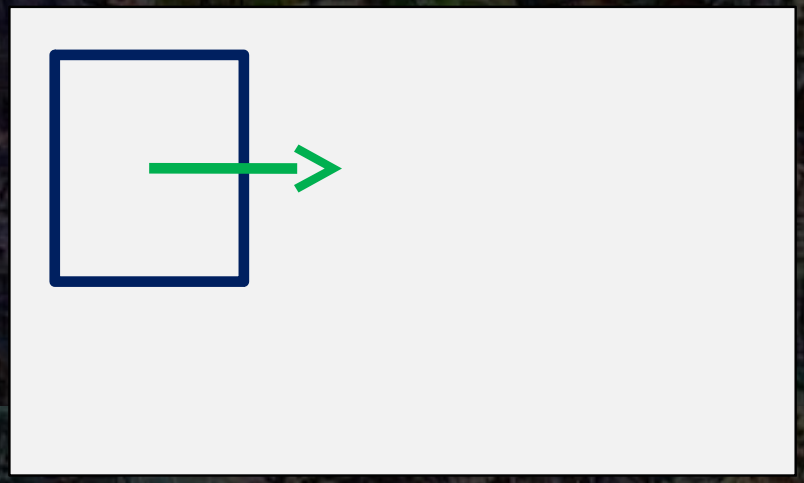
4 Resumen CNN

- Excelente desempeño
- Aprende relación entrada => salida
- Analizan una imagen de resolución fija
  - AlexNet : 227 x 227
- Limitaciones:
  - El objeto debe estar centrado en la imagen
  - Una sóla escala

- Fin primera parte...

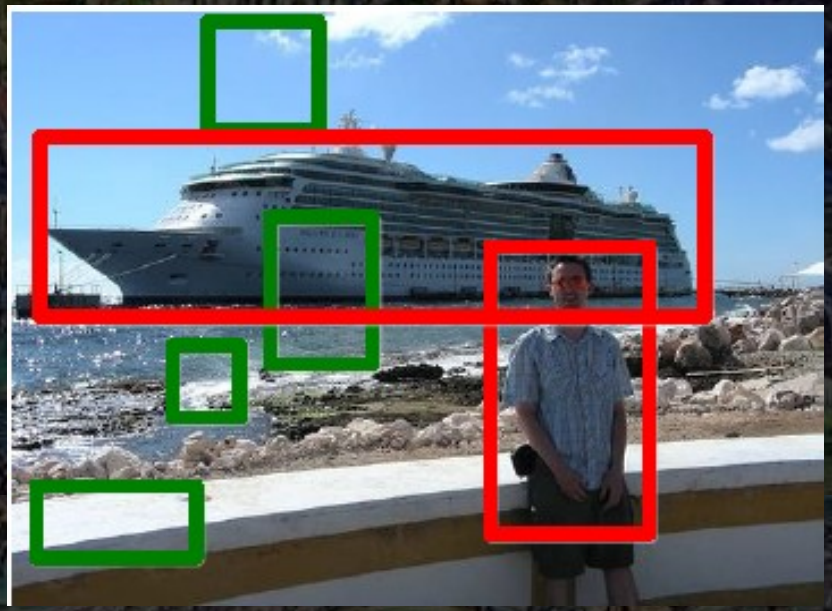
5 Region proposals

- Tema intermedio
- Varias posiciones, escalas => ventana deslizante
- 640 x 480 => 307200
  - Sólo se puede usar con algoritmos muy rápidos



5 Region proposals

- Solución: region proposals
  - Se puede considerar un detector débil de objetos
  - Entregan aprox 1000 ventanas
  - 300 veces menos ventanas que el método deslizante



5 Region proposals

- Puntos en contra
  - Puede eliminar algunas ventanas que contenían objetos
- Puntos a favor
  - Permite usar mejores detectores
  - Mejor MAP al reducir variabilidad de ventanas

5 Region proposals

- Métodos clásicos para calcular region proposals:
  - Selective Search
    - Basado en segmentación
  - BING
    - Detecta patrones de bordes
  - Edgeboxes
    - Detecta patrones de bordes

5 Region proposals

- Métodos clásicos para calcular region proposals:
  - Selective Search
    - Basado en segmentación
  - BING
    - Detecta patrones de bordes
  - Edgeboxes
    - Detecta patrones de bordes

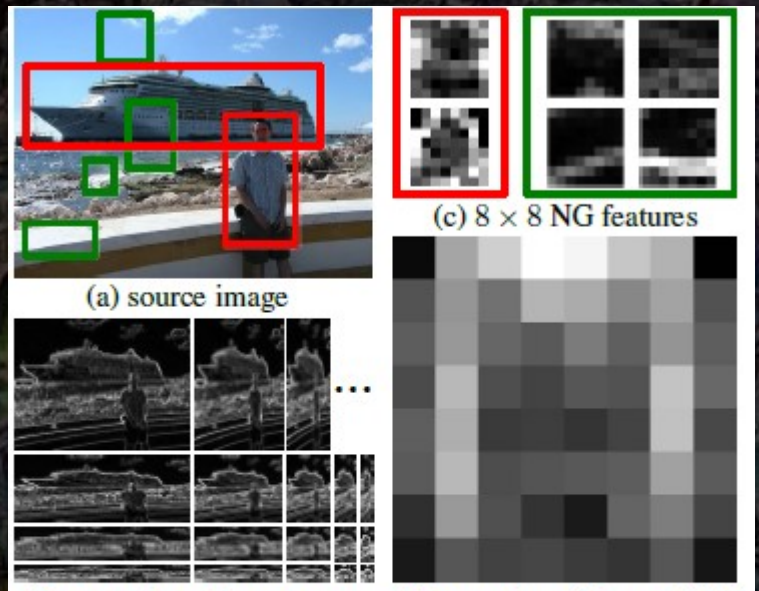
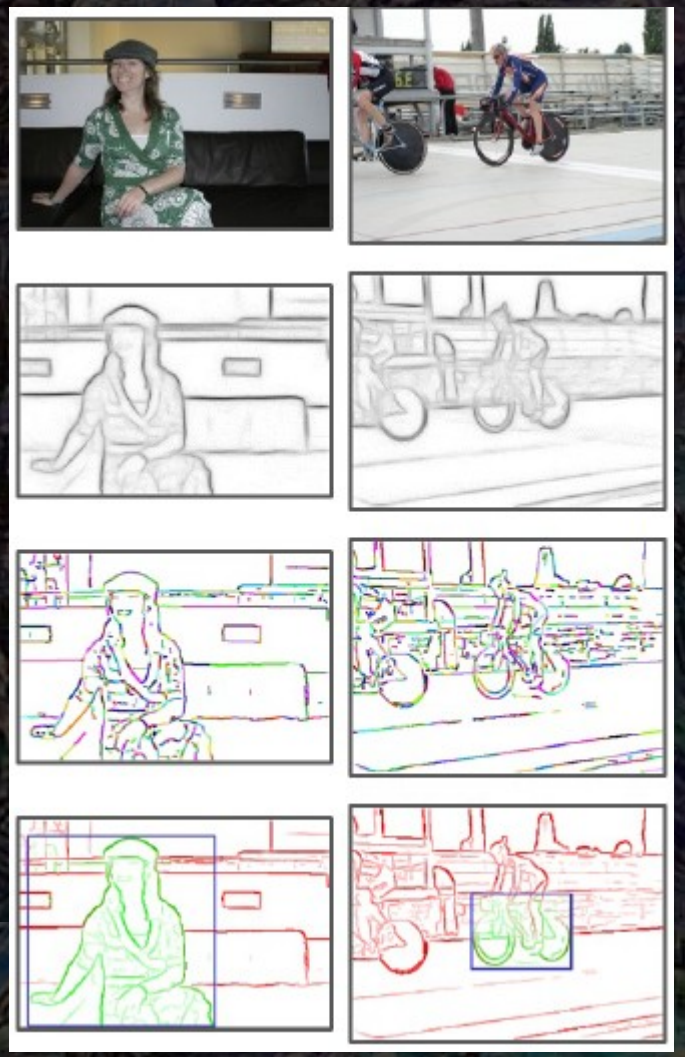


Figure 1. Although object (red) and non-object (green) windows present huge variation in the image space (a), in proper scales and aspect ratios where they correspond to a small fixed size (b), their corresponding normed gradients, *i.e.* a NG feature (c), share strong correlation. We learn a single 64D linear model (d) for selecting object proposals based on their NG features.

5 Region proposals

- Métodos clásicos para calcular region proposals:
  - Selective Search
    - Basado en segmentación
  - BING
    - Detecta patrones de bordes
  - Edgeboxes
    - Detecta patrones de bordes

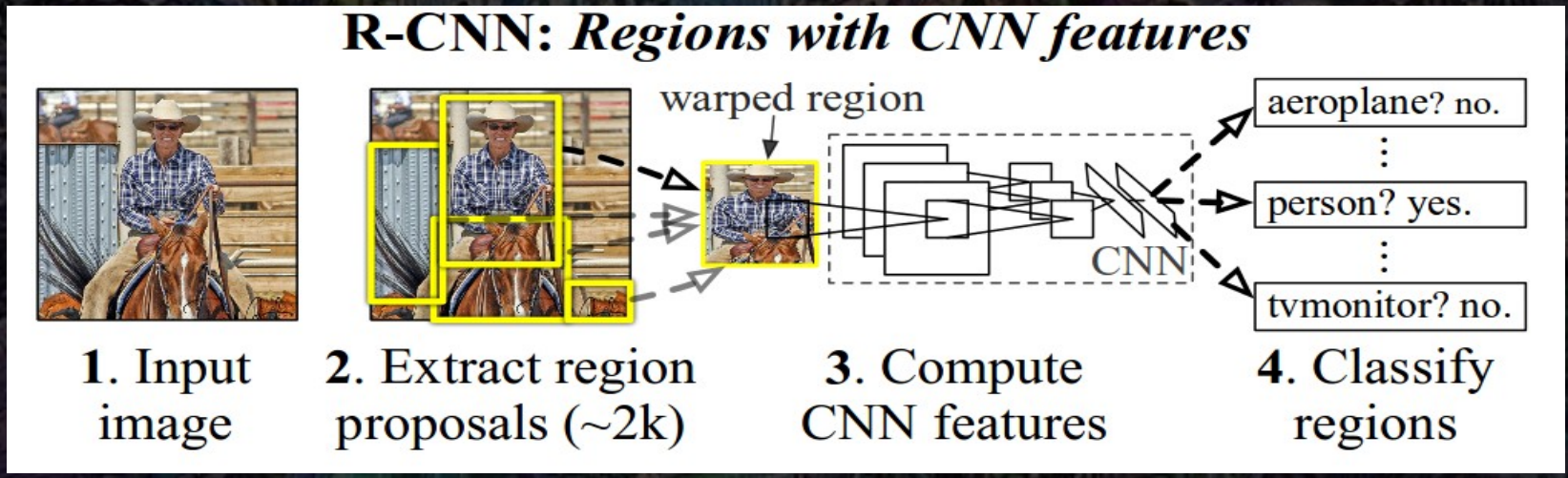


5 Region proposals

- Una vez que se tienen los proposals, se puede aplicar CNN a cada ventana
- Detección rápida y confiable de objetos

6 Arquitecturas de R-CNNs

- R-CNN
  - Usa SelectiveSearch seguido de CNN (AlexNet)
  - CNN se recalcula para cada objeto
  - Viene incluida con Caffe (software)

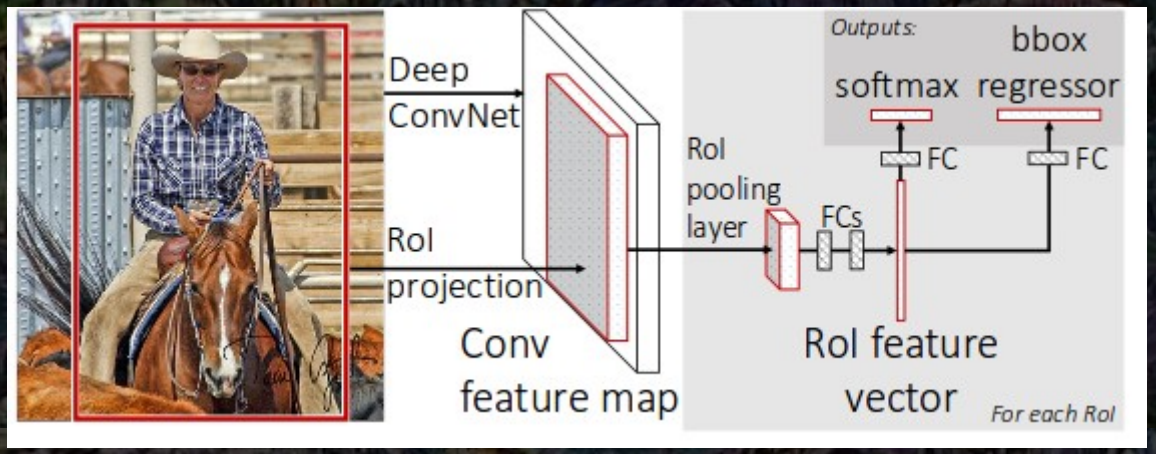


  - ~/Documents/src/caffe/caffe-faster/caffe/examples (faster-R-CNN)$ ipython notebook detection.ipynb

6 Arquitecturas de R-CNNs

- Fast R-CNN
  - Usa VGG16 (mayor que en R-CNN)
  - Capas convolucionales compartidas
  - 213 veces más rápido que R-CNN
  - Mejora el bounding box

http://arxiv.org/abs/1504.08083



6 Arquitecturas de R-CNNs

- Faster-RCNN
  - Estado del arte en deteción de objetos
  - Agrega una Region Proposal Network (RPN)
    - Usa características básicas convolucionales de la red
    - Predice bounding boxes y scores

- http://arxiv.org/abs/1506.01497

6 Arquitecturas de R-CNNs

- Faster-RCNN
  - Se calculan las capas convolucionales
  - Se aplica una red deslizante fully connected de 3x3 pixeles sobre la última capa convolucional
    - Red de regresión (reg)
    - Red de clasificación (cls)
  - La red de clasificación indica si en la ventana de 3x3 corresponde a un proposal
  - La red de regresión mejora la estimación del bounding box del proposal

6 Arquitecturas de R-CNNs

- Faster-RCNN

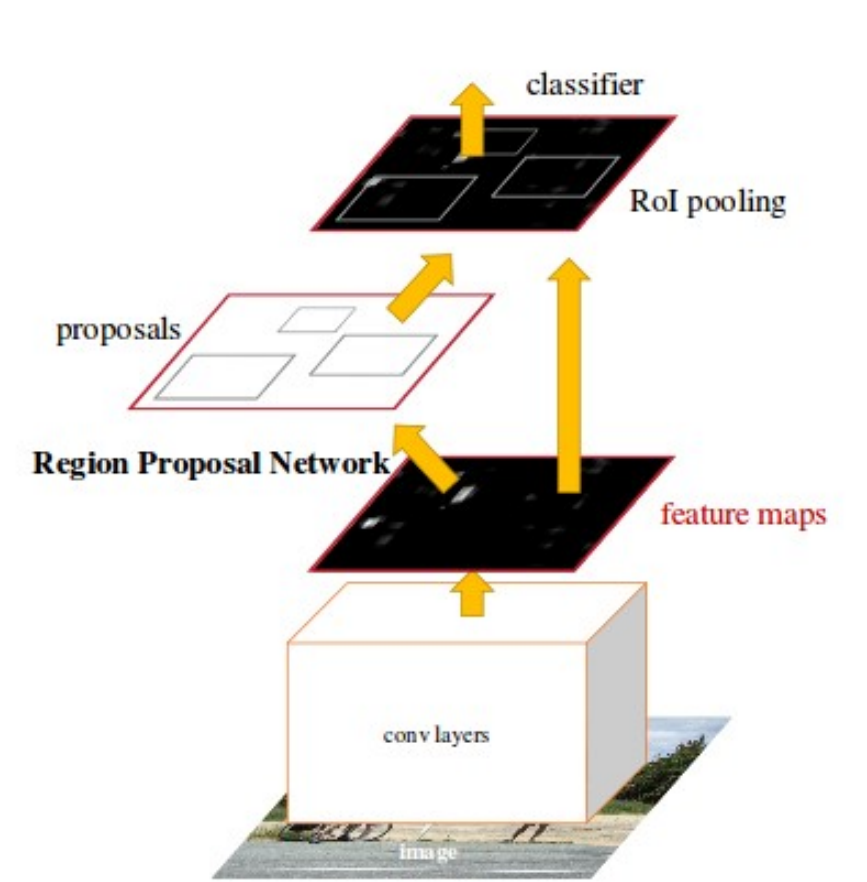


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

6 Arquitecturas de R-CNNs

- Una vez que se tienen los proposal, se usa R-CNN para clasificarlos
- La red R-CNN reutiliza las mismas capas convolucionales usadas por la RPN

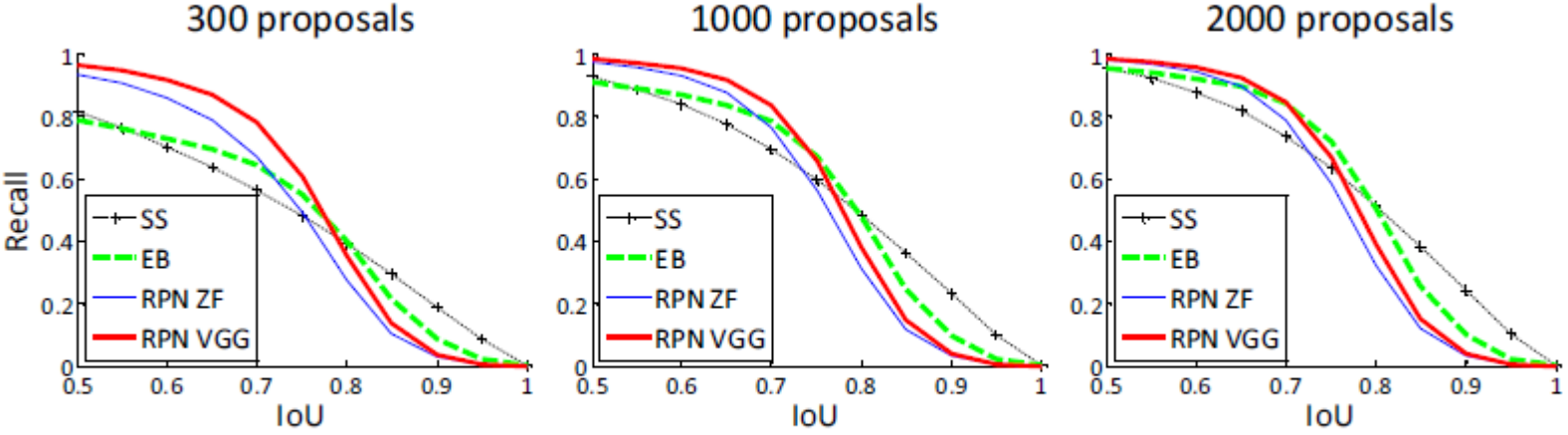


Figure 4: Recall *vs.* IoU overlap ratio on the PASCAL VOC 2007 test set.

6 Resumen R-CNNs

- R-CNN: permiten detectar varios objetos dentro de una imagen
- Nuevas arquitecturas incluyen la generación de region proposals
  - Proposals se entrenan junto con la red CNN

- Fin de la segunda parte

7 Herramientas de software

- Bibliotecas para CNNs
- Hay varias disponibles
  - Caffe (y variantes)
  - MatConvNet (implementado en VLFeat)
  - Theano, Torch, …

7.1 Caffe

- Biblioteca que permite entrenar y ejecutar CNNs
- Escrita en C++ / CUDA
- Puede ser usada mediante python y C++
- Varios algoritmos de CNN implementados

7.1 Caffe - Instalación

- Requiere cosas como
  - Python
  - Protobuf
  - HDF5
  - OpenCV
  - CUDA (Opcional)
  - MATLAB (Opcional)
  - CUDNN (Opcional)

7.2 Caffe – Formatos de archivos

- El modelo se especifica por una serie de capas (layers), que pueden ser de distintos tipos:
- Vision layers
- Loss layers
- Activation layers (ReLU)
- Data layers
- Common layers

7.2 Caffe – Formatos de archivos

- Ejemplo capa visión: bottom → operación → top

```
name: "CaffeNet"
input: "data"
input_dim: 10
input_dim: 3
input_dim: 227
input_dim: 227
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  convolution_param {
    num_output: 96
    kernel_size: 11
    stride: 4
  }
}
```

7.2 Caffe – Formatos de archivos

- Ejemplo de .prototxt (protobuf)

```
name: "AlexNet"
input: "data"
input_shape {
  dim: 10
  dim: 3
  dim: 227
  dim: 227
}
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution_param {
    num_output: 96
    kernel_size: 11
    stride: 4
  }
}
layer {
  name: "relu1"
  type: "ReLU"
  bottom: "conv1"
  top: "conv1"
}
layer {
  name: "norm1"
  type: "LRN"
```

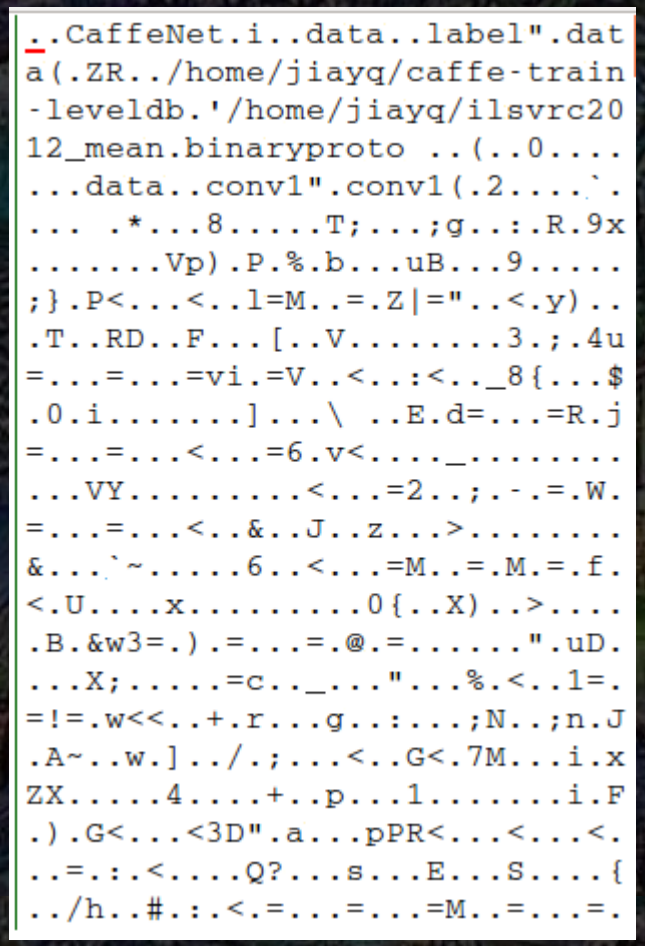
```
layer {
  name: "norm1"
  type: "LRN"
  bottom: "conv1"
  top: "norm1"
  lrn_param {
    local_size: 5
    alpha: 0.0001
    beta: 0.75
  }
}
layer {
  name: "pool1"
  type: "Pooling"
  bottom: "norm1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "conv2"
  type: "Convolution"
  bottom: "pool1"
  top: "conv2"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
```

...

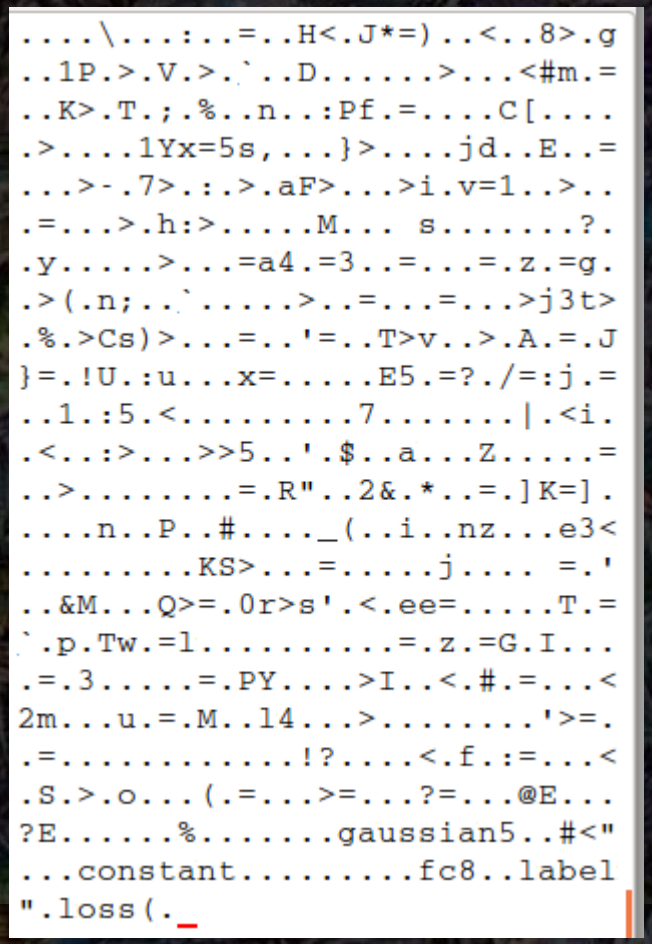
```
  type: "ReLU"
  bottom: "fc7"
  top: "fc7"
}
layer {
  name: "drop7"
  type: "Dropout"
  bottom: "fc7"
  top: "fc7"
  dropout_param {
    dropout_ratio: 0.5
  }
}
layer {
  name: "fc8"
  type: "InnerProduct"
  bottom: "fc7"
  top: "fc8"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  inner_product_param {
    num_output: 1000
  }
}
layer {
  name: "prob"
  type: "Softmax"
  bottom: "fc8"
  top: "prob"
}
```

7.2 Caffe – Formatos de archivos

- .caffemodel (parámetros): Binario



...



7.3 Caffe - Ejecución

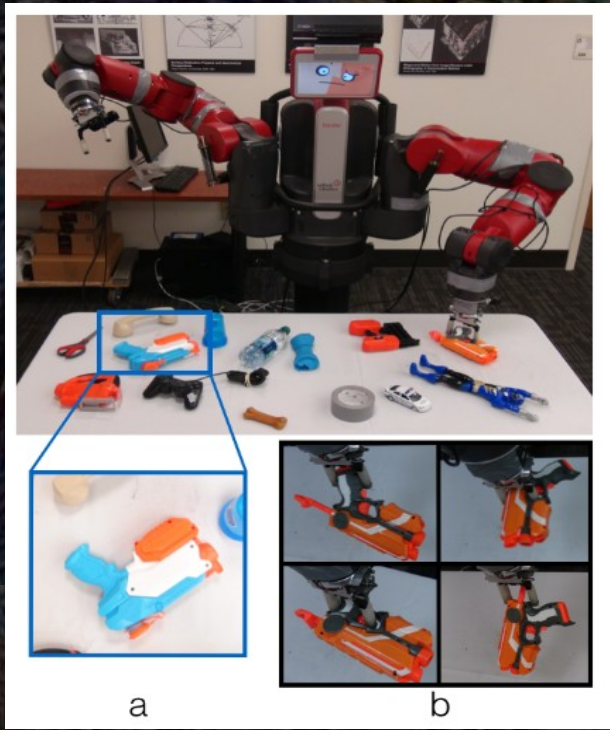
- Hay varias formas de ejecutar caffe
- La típica forma es usando python
- Una forma interactiva (tutoriales) es usando ipython notebook
- Ejemplo
  - ipython notebook 00-classification.ipynb
  - Ipython notebook detection.ipynb
    - En /home/ploncomi/Documents/src/caffe/caffe-faster/caffe/examples

8 Aplicaciones

- Reconocimiento de acciones
  - Donahue et al 2014: Usan inicialmente canales RGB y de flujo óptico
- Captioning de imágenes y videos
  - CNN + LSTM
- Image denoising
  - Usando autoencoders
- Similarity learning

8 Aplicaciones

- Manipulación de objetos
  - Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours
  - Aprende a manipular sin usar detector de objetos



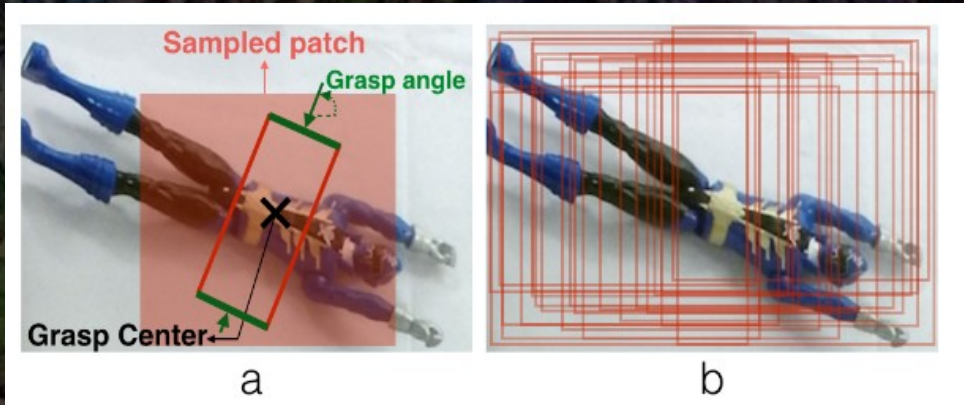
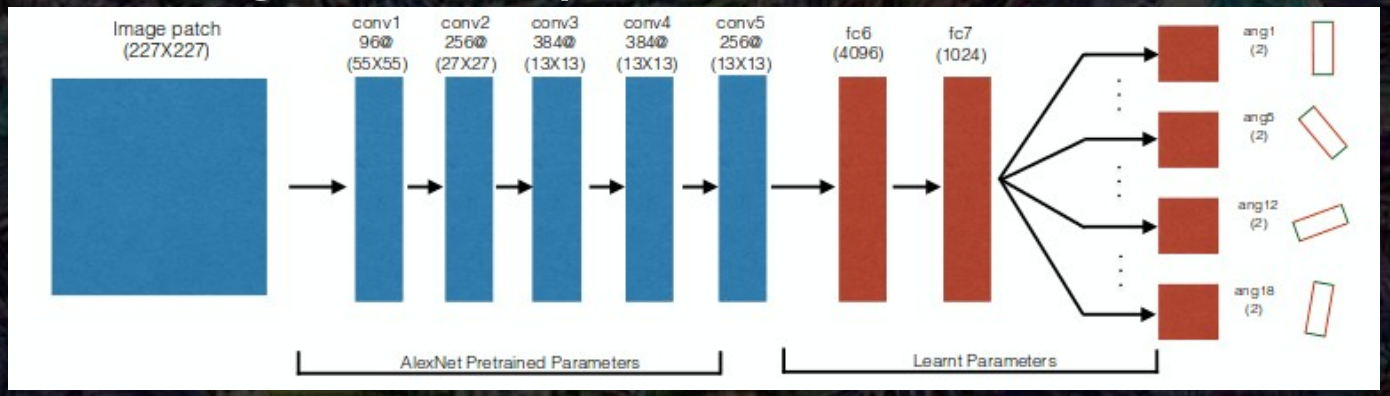


Fig. 3. (a) We use 1.5 times the gripper size image patch to predict the grasp-ability of a location and the angle at which it can be grasped. Visualization for showing the grasp location and the angle of gripper for grasping is derived from [8]. (b) At test time we sample patches at different positions and choose the top graspable location and corresponding gripper angle.

8 Aplicaciones

- Manipulación de objetos
  - Supersizing Self-supervision



8 Aplicaciones

- Visual question answering
  - Basado en captioning
  - Open answer v/s multiple choice

How many pickles are on the plate?	1 1 1	1 1 1
What is the shape of the plate?	circle round round	circle round round

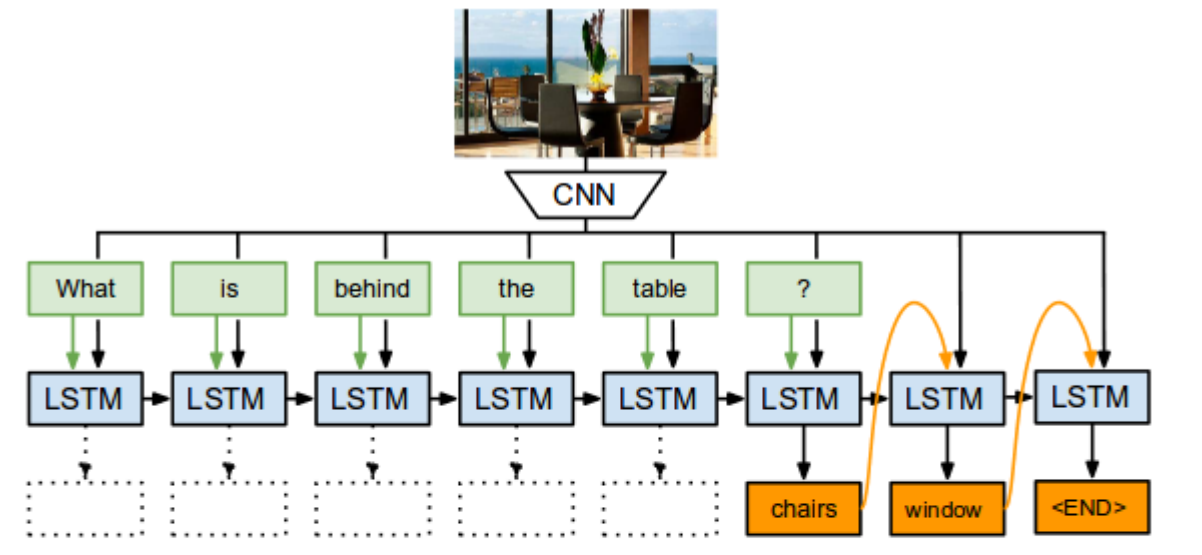
What does the sign say?	stop stop stop	stop stop yield
What shape is this sign?	octagon octagon octagon	diamond octagon round

8 Aplicaciones

- Visual question answering
  - Neural-Image-QA
  - “Ask your neurons: A neural-based approach..”
  - El sistema aprende a representar la relación entre las palabras y la imagen
  - Es decir, no se usan detectores de objetos

8 Aplicaciones

- Visual question answering
- Arquitectura usa:
  - Red recurrente LSTM
  - Features de CNN

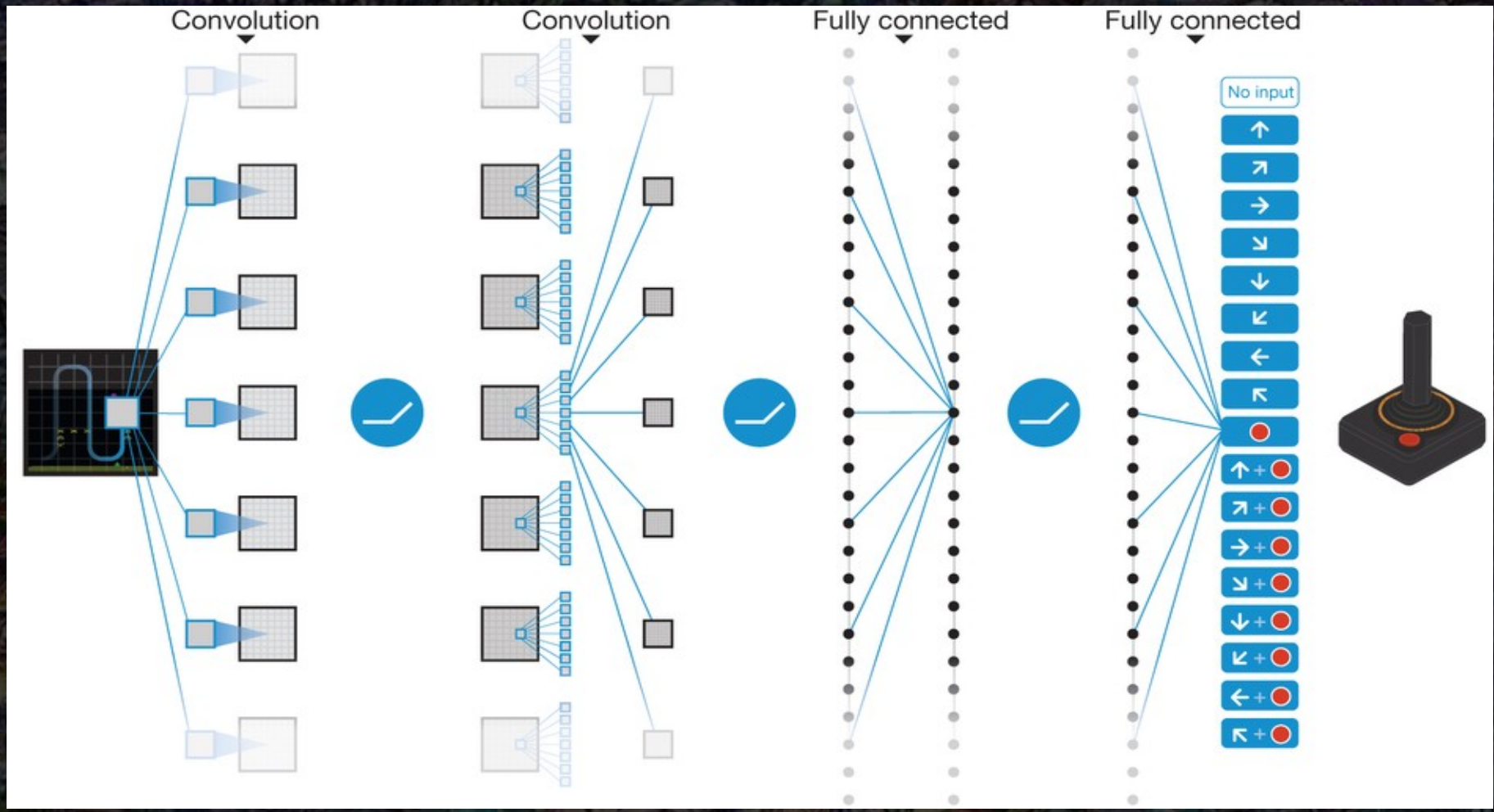


8 Aplicaciones

- Aprendizaje reforzado
  - Deep Q Networks (recompensa)
    - Función Q modelada con red neuronal
  - Ej: algoritmo que juega Atari
    - Recibe imágenes en escalas de gris de 84x84
    - Usa 4 frames (distintos instantes de tiempo)

8 Aplicaciones

- Resultado del entrenamiento:



9 Resumen

- Se ha visto la teoría detrás de las redes tipo CNN, R-CNN
- Se ha visto un conjunto de arquitecturas útiles
- Se ha mostrado una herramienta (Caffe) que permite usar CNNs en la práctica
- Se ha mostrado brevemente un conjunto de aplicaciones que son áreas de investigación actual relacionadas con CNN
- Códigos disponibles :)

10 Links

- (Godfellow) http://www.deeplearningbook.org/
- (ResNet) http://research.microsoft.com/en-us/um/people/kahe/ilsvrc15/ilsvrc2015_deep_residual_learning_kaiminghe.pdf
- (Faster R-CNN) http://arxiv.org/pdf/1506.01497v3.pdf
- Benchmark segmentación semántica
  - https://www.cityscapes-dataset.com/benchmarks/#scene-labeling-task

10 Links

- Benchmark reconocimiento objetos http://www.image-net.org/
- (VQA) http://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Malinowski_Ask_Your_Neurons_ICCV_2015_paper.pdf
- (VQA) http://www.visualqa.org/
- Deep-Q-Networks
  - https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf