Lecture 05

Introducción a redes neuronales convolucionales (CNNs)

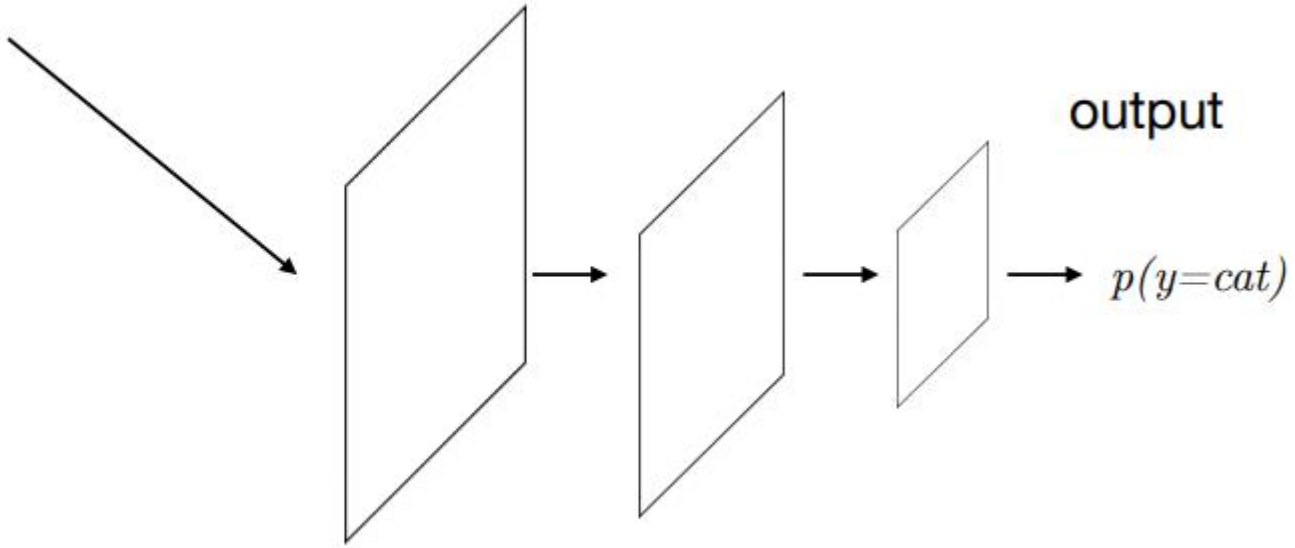
Aplicaciones Comunes de las CNN

1. **¿Qué pueden hacer las CNNs?**
2. Clasificación de imágenes
3. Conceptos básicos de CNNs
4. Filtros convolucionales y pesos compartidos
5. Correlación cruzada vs convolución
6. CNNs y backpropagation (retropropagación)
7. Arquitectura de las CNNs
8. Lo que pueden ver las CNN
9. CNNs en PyTorch

CNN para clasificación de imágenes

Image Source:
twitter.com%2Fcats&psig=AOvVaw30_o-PCM-K21DiMAJQimQ4&ust=1553887775741551

Image Source: https://www.pinterest.com/pin/244742560974520446



Detección de objetos

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 779-788).

Segmentación de Objetos

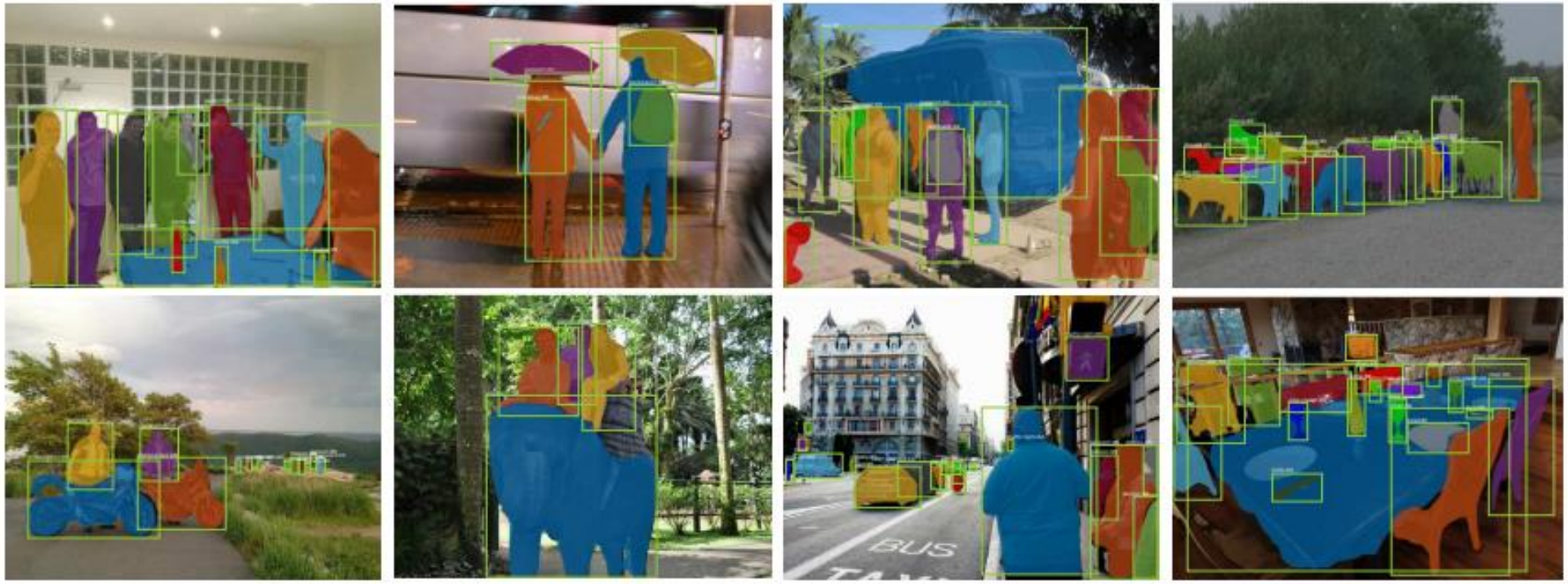
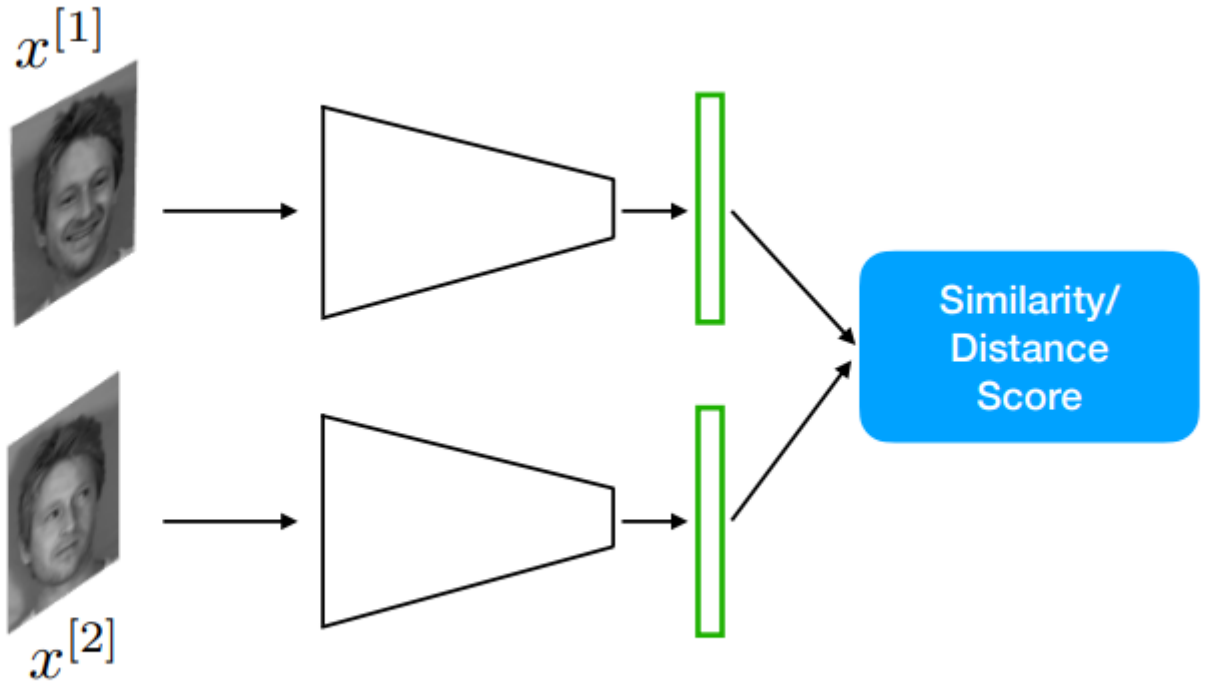


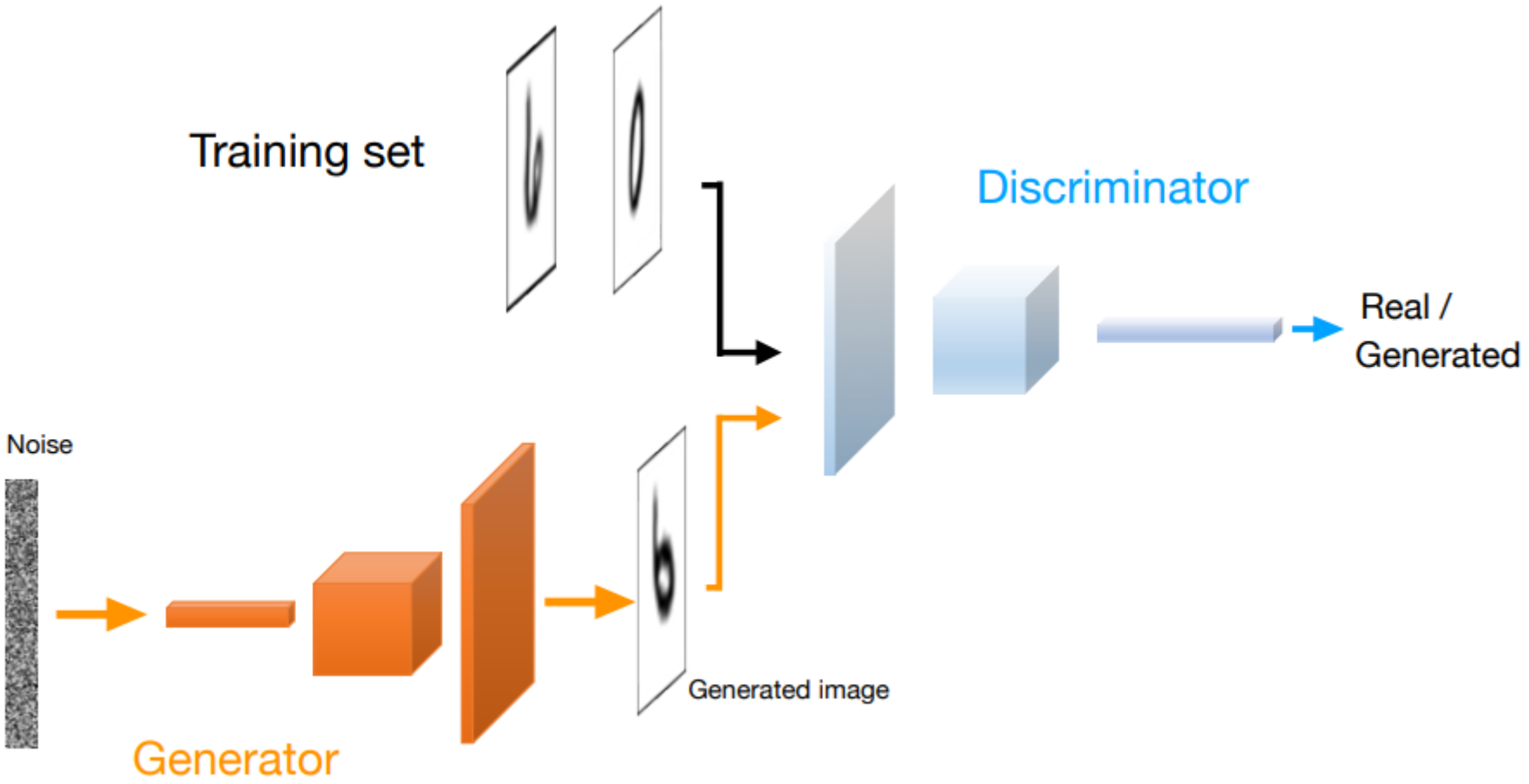
Figure 2. **Mask R-CNN** results on the COCO test set. These results are based on ResNet-101 [15], achieving a *mask* AP of 35.7 and running at 5 fps. Masks are shown in color, and bounding box, category, and confidences are also shown.

He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask R-CNN." In Proceedings of the IEEE International Conference on Computer Vision, pp. 2961-2969. 2017.

Reconocimiento facial



Síntesis de Imágenes



¿Por qué la visión por computador es (era) difícil?: Retos con la clasificación de imágenes

1. ¿Qué pueden hacer las CNNs?
2. **Clasificación de imágenes**
3. Conceptos básicos de CNNs
4. Filtros convolucionales y pesos compartidos
5. Correlación cruzada vs convolución
6. CNNs y backpropagation (retropropagación)
7. Arquitectura de las CNNs
8. Lo que pueden ver las CNN
9. CNNs en PyTorch

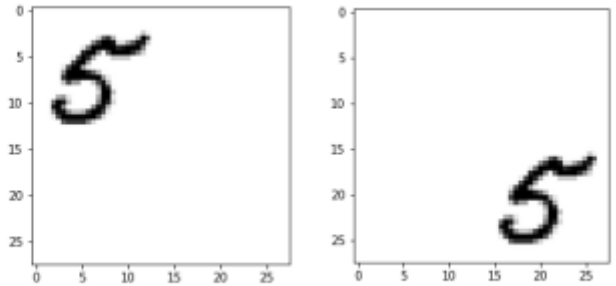
¿Por qué la clasificación de imágenes es difícil?

Diferente iluminación, contraste, puntos de vista, etc.

Image Source: twitter.com%2Fcats&psig=AOvVaw30_o-PCM-K21DiMAJQimQ4&ust=1553887775741551

Image Source: https://www.123rf.com/photo_76714328_side-view-of-tabby-cat-face-over-white.html

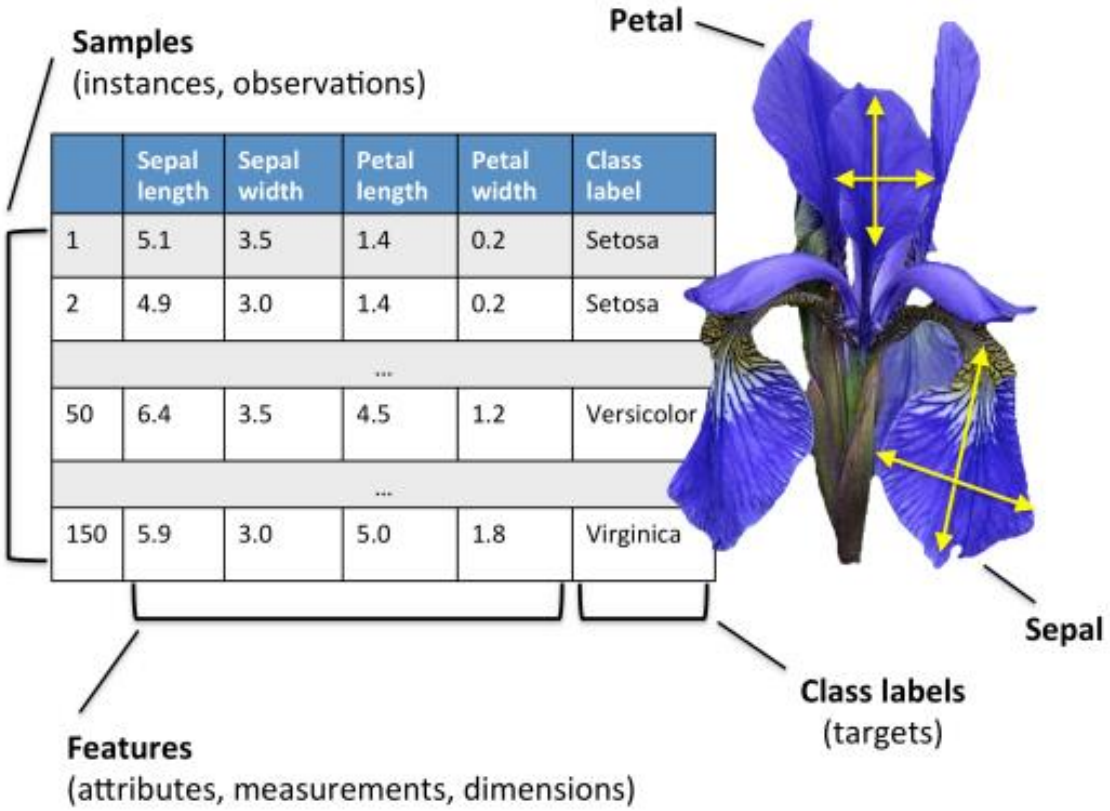
O in



Esto es difícil para métodos tradicionales como perceptrones multicapa porque la predicción está basada en la suma de la intensidad de los pixeles

Enfoques tradicionales

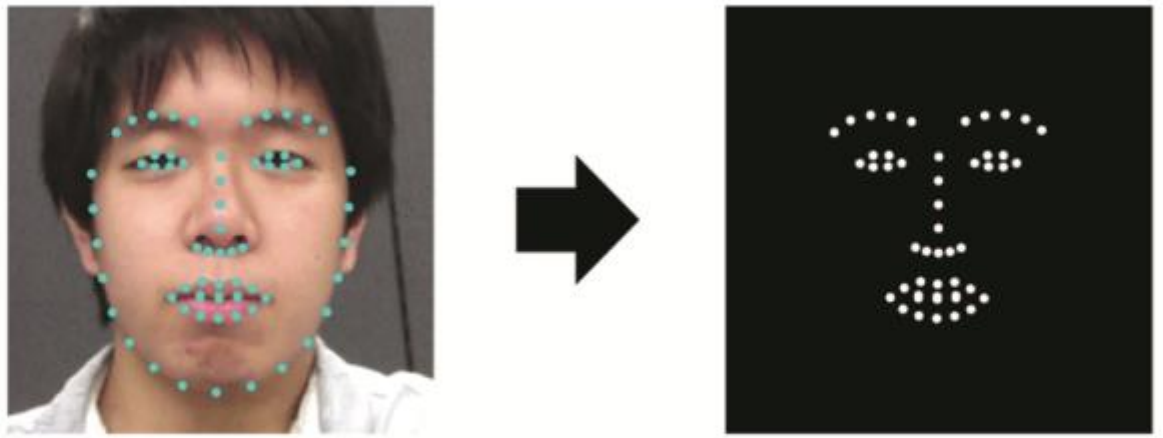
a) Utilizar características diseñadas por humanos (hand crafted features)



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

Enfoques tradicionales

a) Utilizar características diseñadas por humanos (hand crafted features)



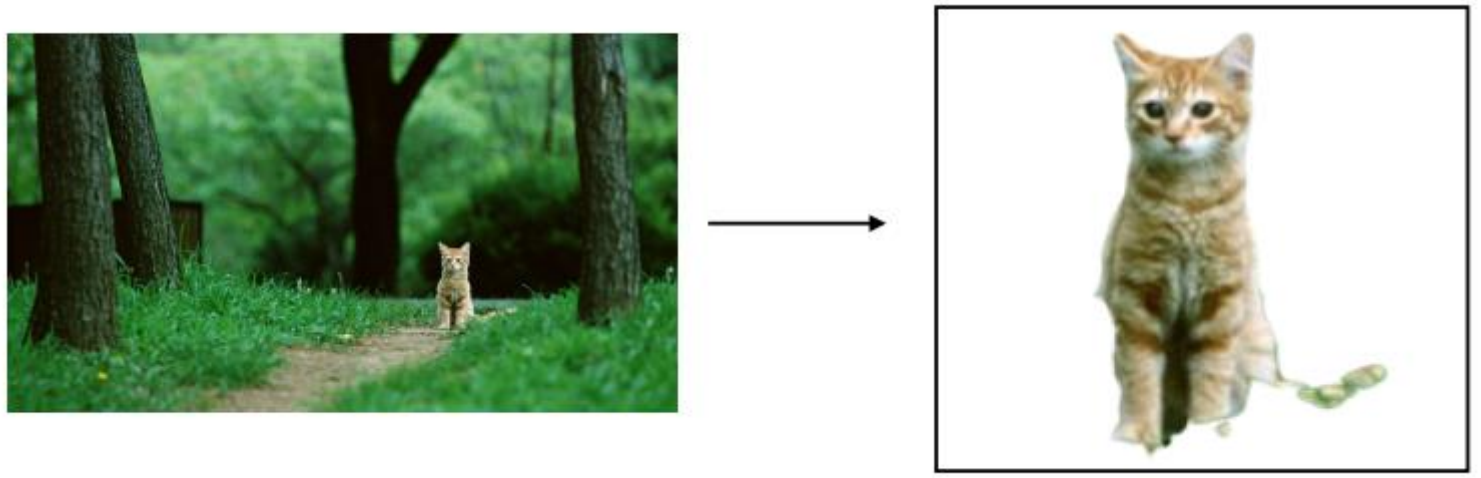
(a) Detected facial keypoints

(b) Facial organ keypoints

Sasaki, K., Hashimoto, M., & Nagata, N. (2016). Person Invariant Classification of Subtle Facial Expressions Using Coded Movement Direction of Keypoints. In Video Analytics. Face and Facial Expression Recognition and Audience Measurement (pp. 61-72). Springer, Cham.

Enfoques tradicionales

b) Preprocesar imágenes (centrar, recortar, etc.)

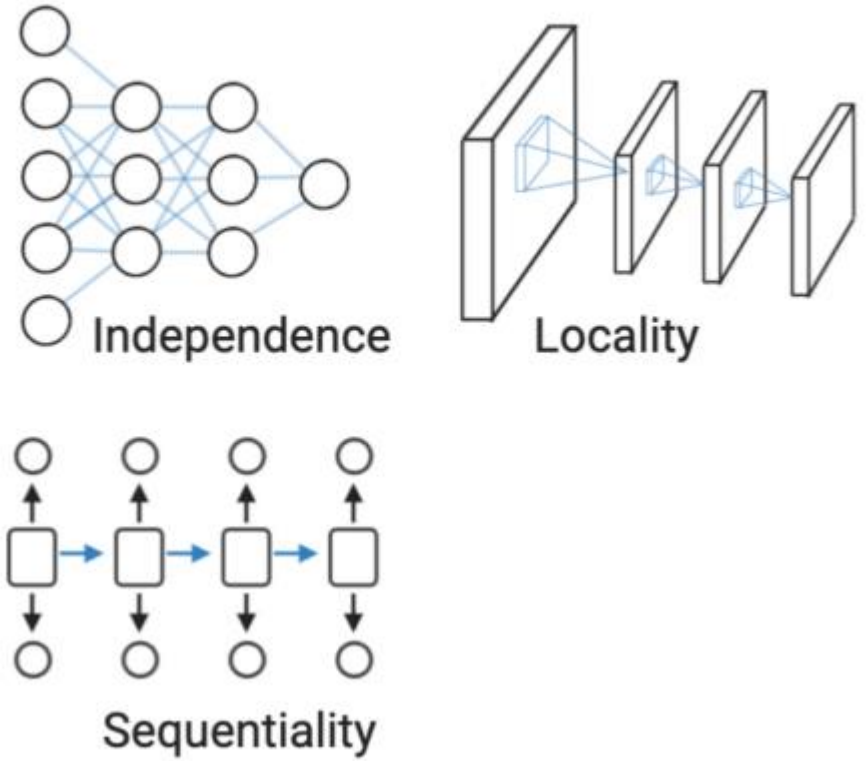


Fuente de imagen: https://www.tokkoro.com/2827328-cat-animals-nature-feline-park-green-trees-grass.html

¿Por qué la visión por computador es (era) difícil?: Retos con la clasificación de imágenes

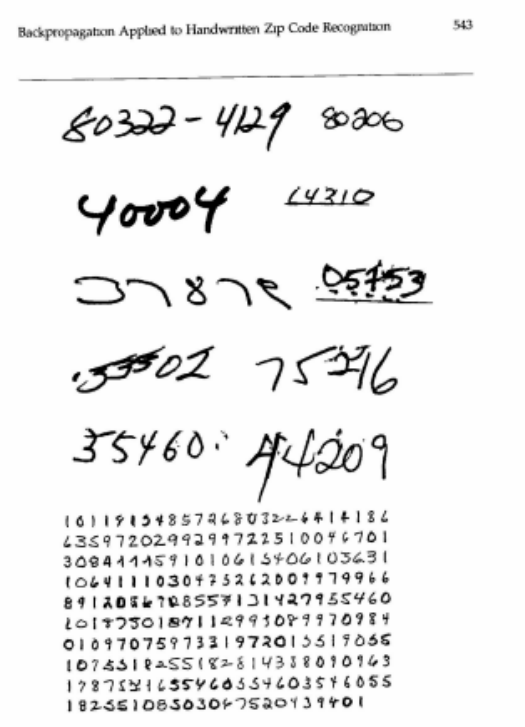
1. ¿Qué pueden hacer las CNNs?
2. Clasificación de imágenes
3. **Conceptos básicos de CNNs**
4. Filtros convolucionales y pesos compartidos
5. Correlación cruzada vs convolución
6. CNNs y backpropagation (retropropagación)
7. Arquitectura de las CNNs
8. Lo que pueden ver las CNN
9. CNNs en PyTorch

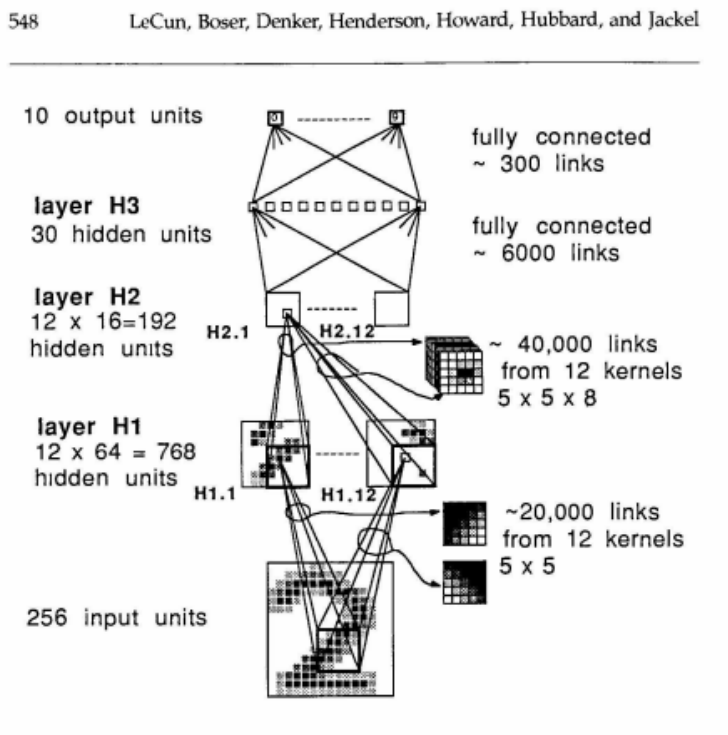
Relational Inductive Biases



https://sgfin.github.io/2020/06/22/Induction-Intro/

Redes neuronales convolucionales





Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel: Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, 1(4):541-551, Winter 1989.

Redes neuronales convolucionales (CNNs)

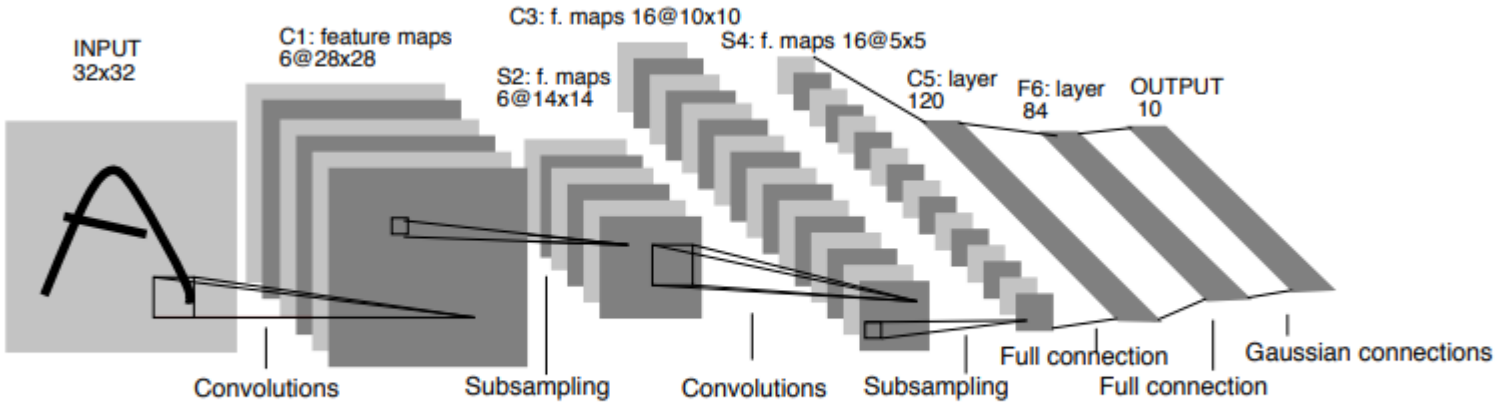
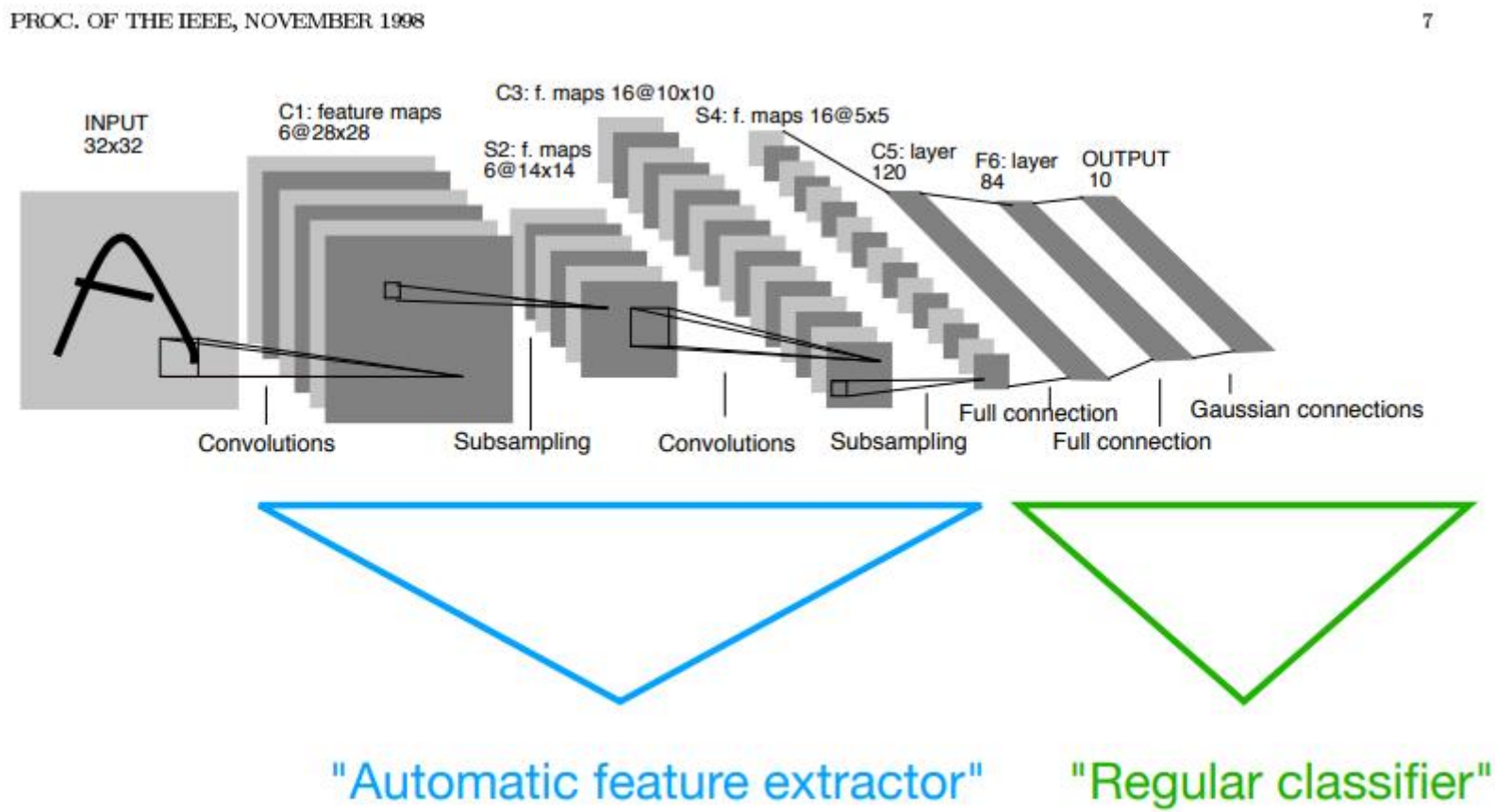


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Yann LeCun, Léon Bottou, Yoshua Bengio and Patrick Haffner: Gradient Based Learning Applied to Document Recognition, Proceedings of IEEE, 86(11):2278–2324, 1998.

Capas ocultas



Capas ocultas

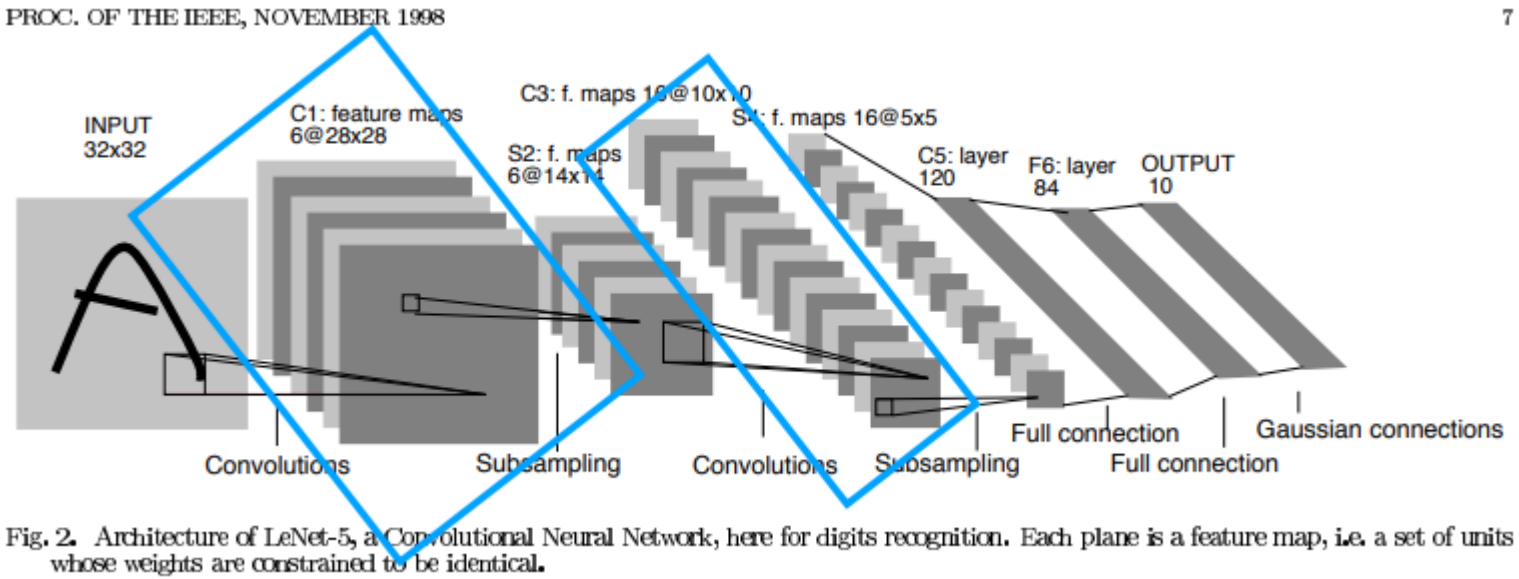
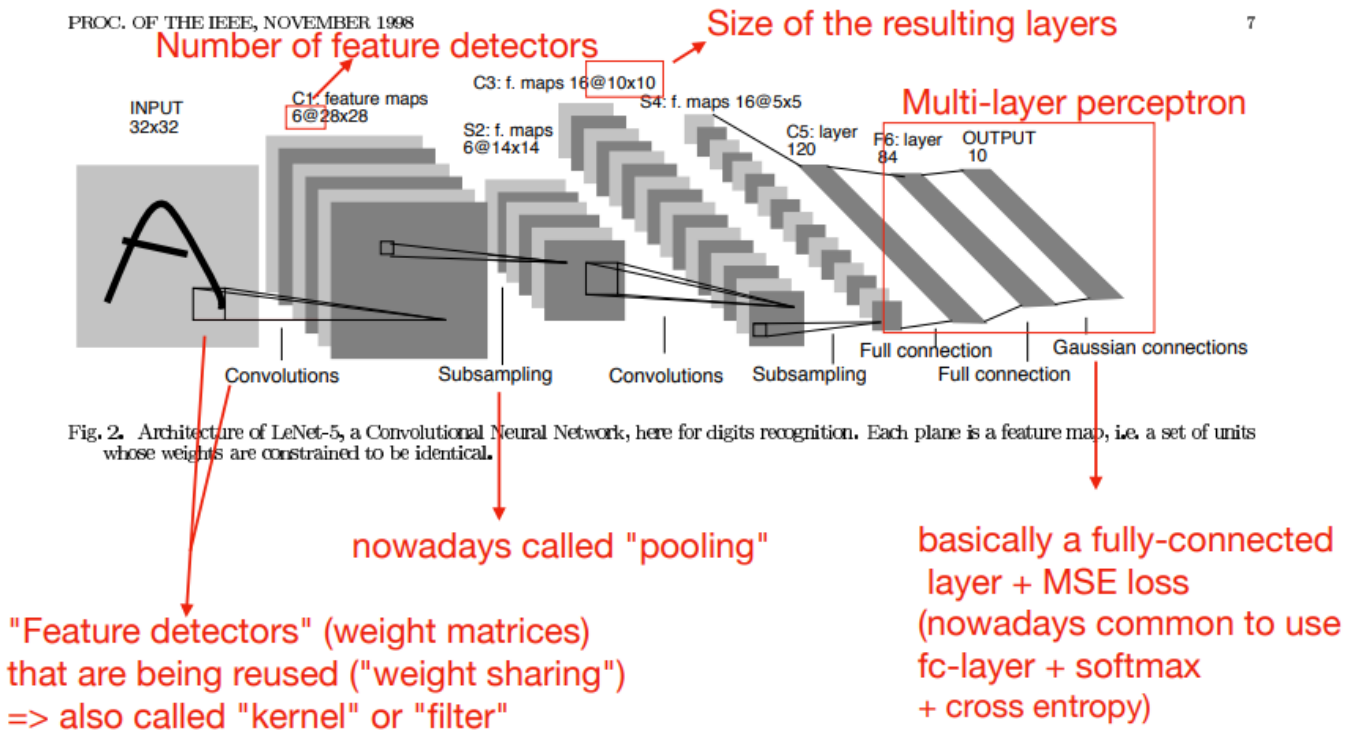


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Cada "grupo" de mapas de características representa una capa oculta en la red neuronal.

Contando las capas de FC, esta red tiene 5 capas

Redes neuronales convolucionales (CNNs)



Yann LeCun, Léon Bottou, Yoshua Bengio and Patrick Haffner: Gradient Based Learning Applied to Document Recognition, Proceedings of IEEE, 86(11):2278–2324, 1998.

Conceptos principales detrás de las redes neuronales convolucionales

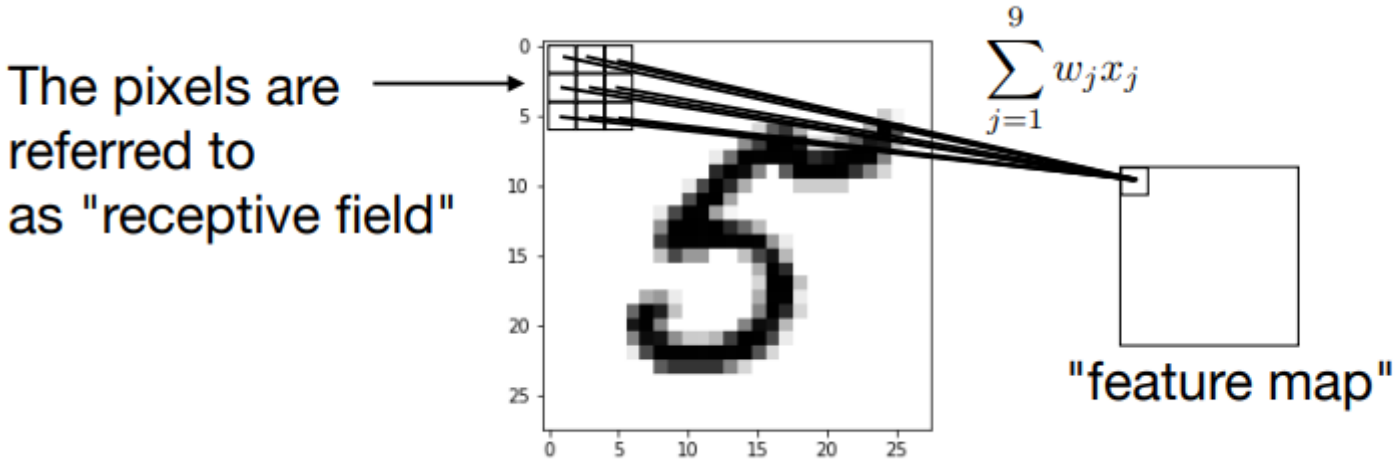
- **Conectividad Escasa:** Un solo elemento en el mapa de características está conectado a solo un pequeño parche de píxeles. (Esto es muy diferente de conectarse a la imagen de entrada completa, en el caso de perceptrones multicapa).
- **Parámetros compartidos:** Se utilizan los mismos pesos para diferentes parches de la imagen de entrada.
- **Muchas capas:** Combinación de patrones locales extraídos con patrones globalesUna mirada más cercana a la capa convolucional

Una mirada más cercana a la capa convolucional

1. ¿Qué pueden hacer las CNNs?
2. Clasificación de imágenes
3. Conceptos básicos de CNNs
4. **Filtros convolucionales y pesos compartidos**
5. Correlación cruzada vs convolución
6. CNNs y backpropagation (retropropagación)
7. Arquitectura de las CNNs
8. Lo que pueden ver las CNN
9. CNNs en PyTorch

Pesos compartidos

Un "detector de características" (filtro, núcleo, kernel) se desliza sobre las entradas para generar un mapa de características

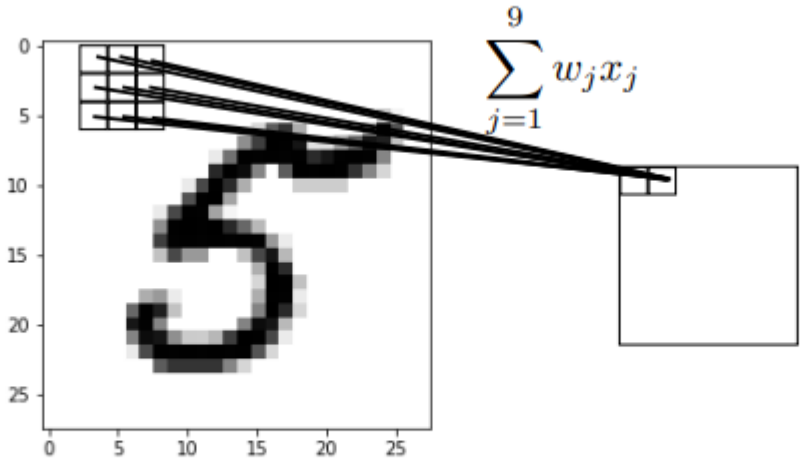


Base lógica: un detector de características que funciona bien en una región también puede funcionar bien en otra región

Además, es una buena reducción de parámetros para adaptarse

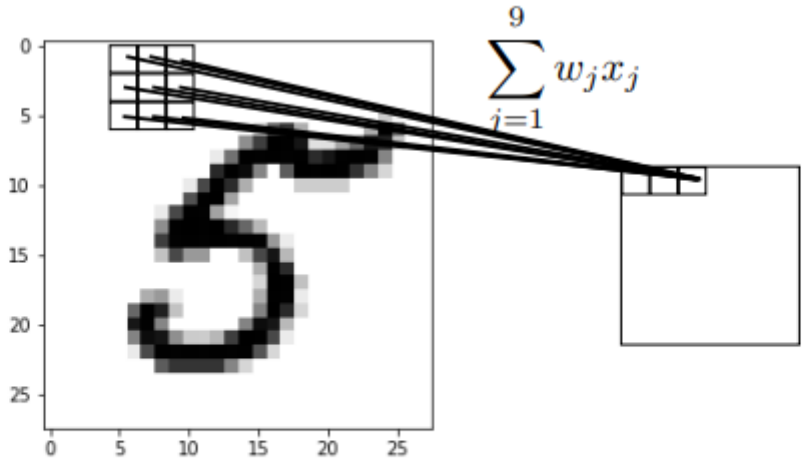
Pesos compartidos

Un "detector de características" (filtro, núcleo) se desliza sobre las entradas para generar un mapa de características



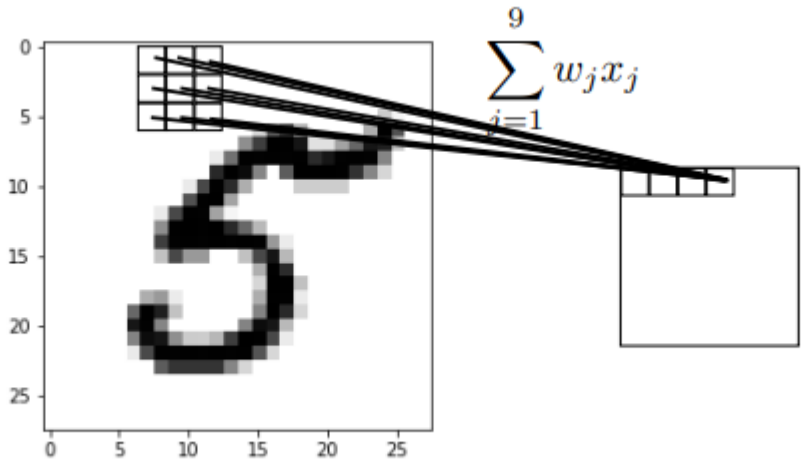
Pesos compartidos

Un "detector de características" (filtro, núcleo) se desliza sobre las entradas para generar un mapa de características



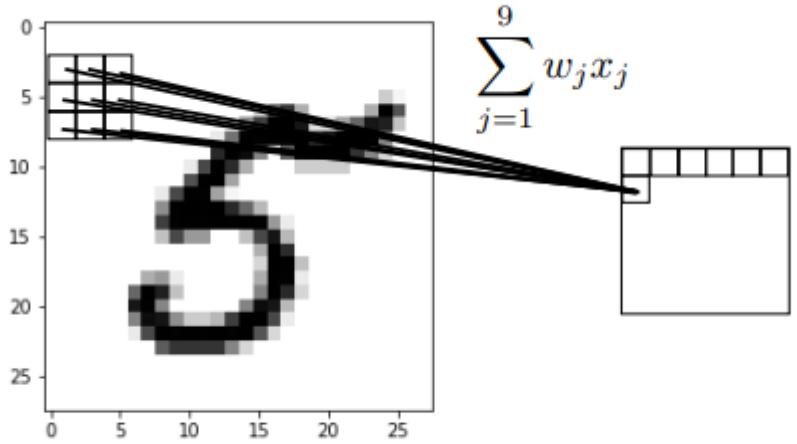
Pesos compartidos

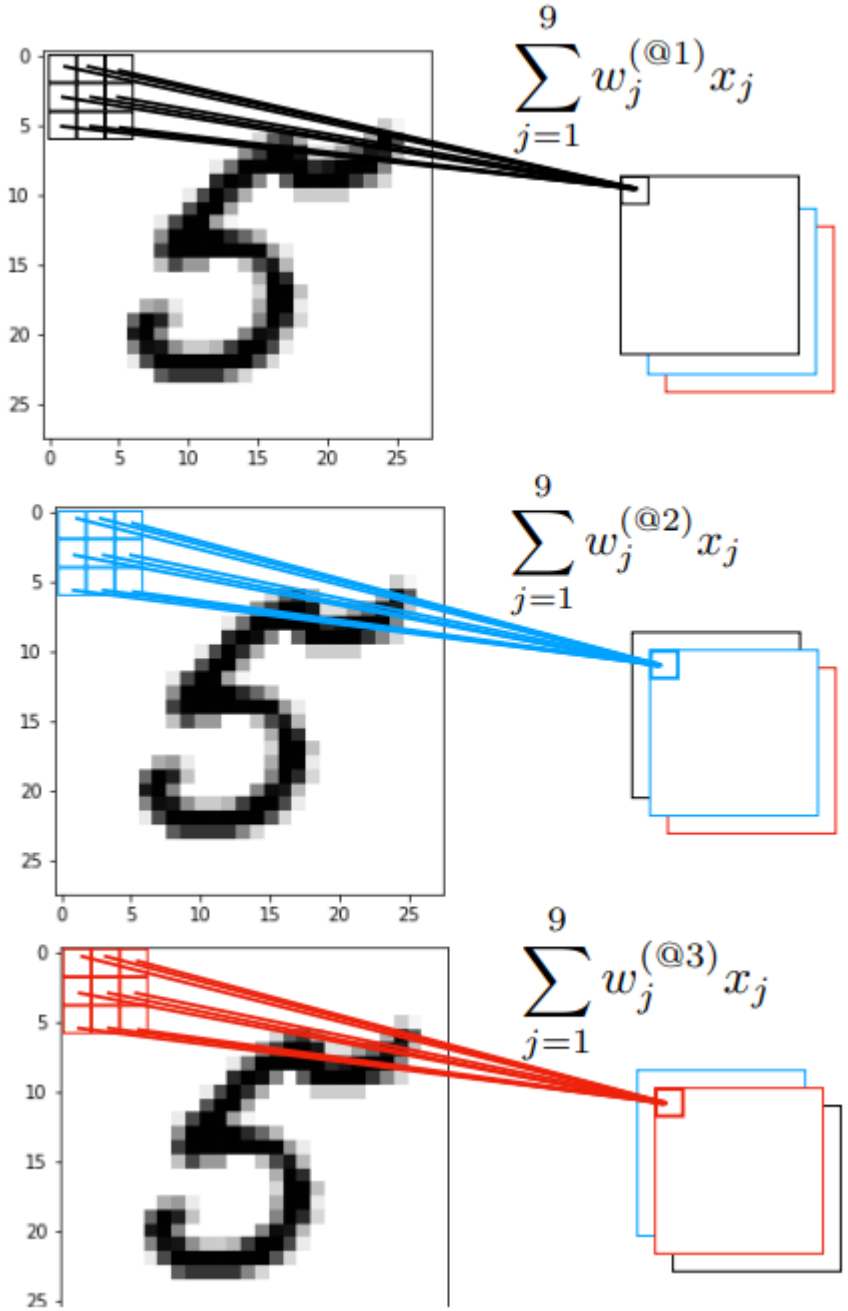
Un "detector de características" (filtro, núcleo) se desliza sobre las entradas para generar un mapa de características



Pesos compartidos

Un "detector de características" (filtro, núcleo) se desliza sobre las entradas para generar un mapa de características

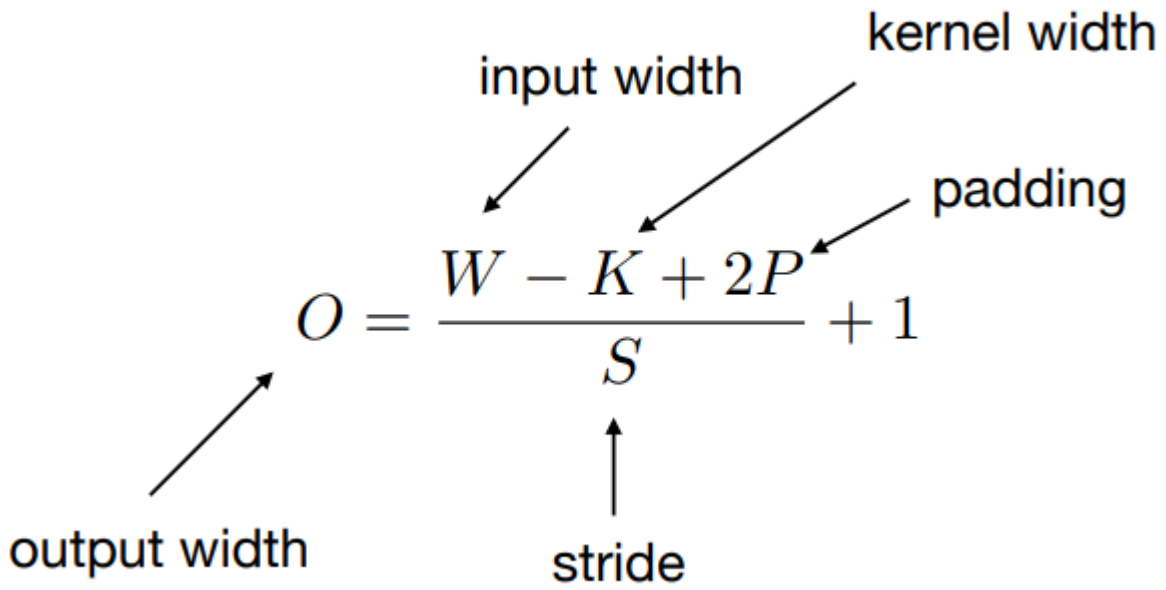




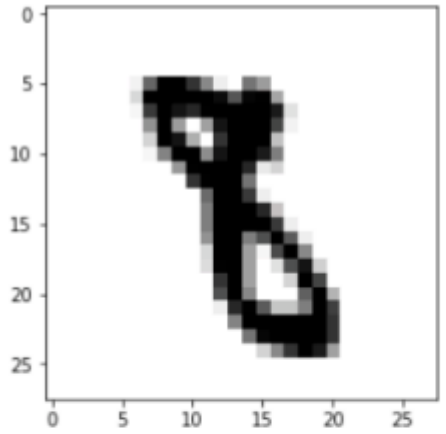
Se utilizan múltiples "detectores de características" (kernels) para crear múltiples mapas de características

Tamaño antes y después de las convoluciones

Tamaño del mapa de características:



Dimensiones del Kernel y Parámetros Entrenables




```
a.shape
```

```
(1, 28, 28)
```

```
import torch
```

```
conv = torch.nn.Conv2d(in_channels=1,
                       out_channels=8,
                       kernel_size=(5, 5),
                       stride=(1, 1))
```

```
conv.weight.size()
```

```
torch.Size([8, 1, 5, 5])
```

```
conv.bias.size()
```

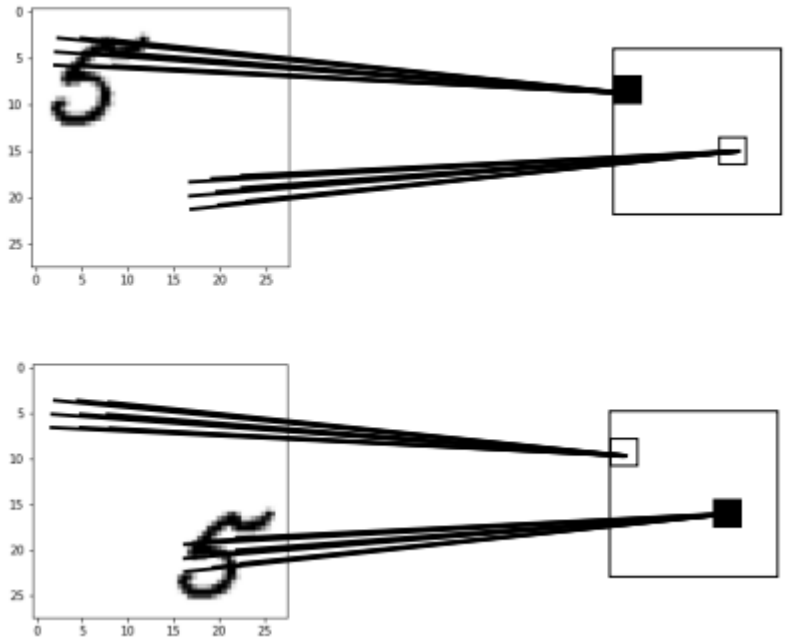
```
torch.Size([8])
```

Para una imagen en escala de grises con un detector de características de 5x5 (kernel), tenemos las siguientes dimensiones (número de parámetros para aprender)

¿Cuál creen que es el tamaño de salida de esta imagen de 28x28?

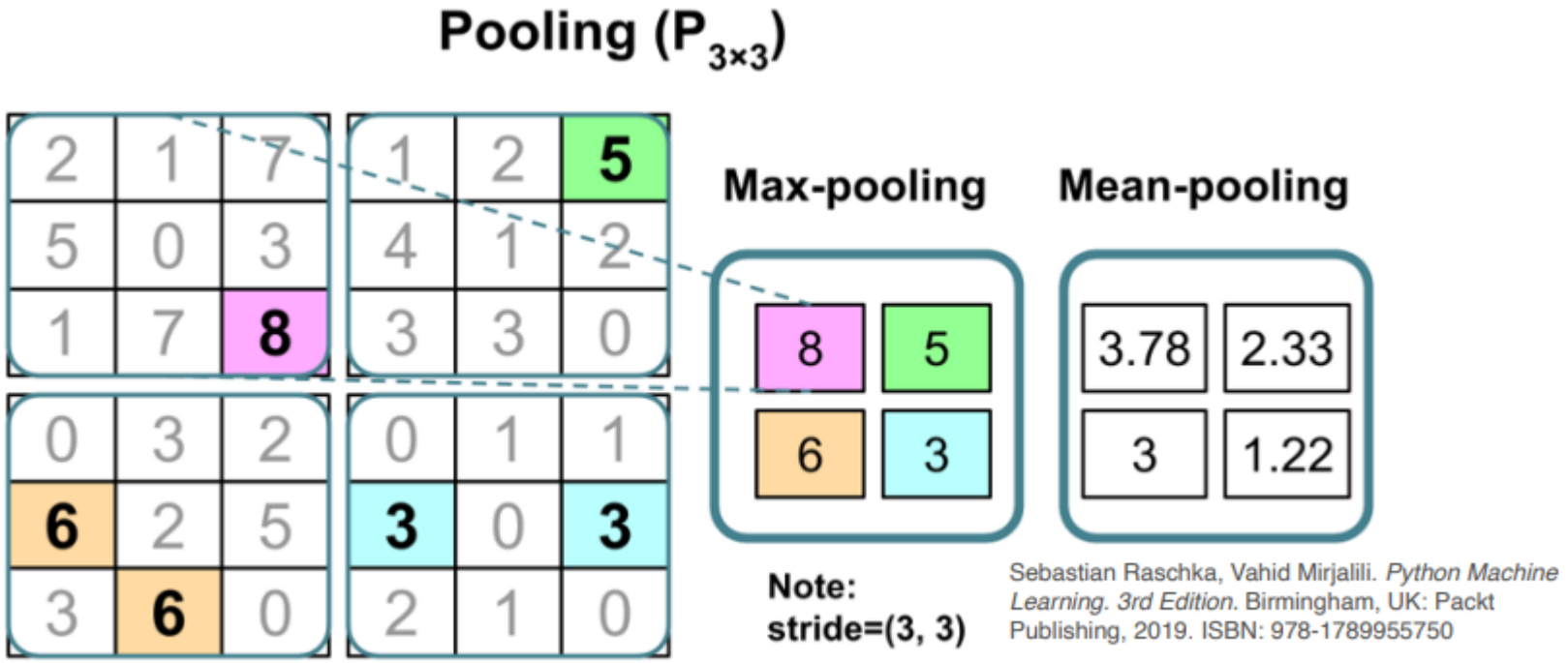
CNNs y traducción/Rotación/Invarianza de escala

Tenga en cuenta que las CNNs no son realmente invariantes a la escala, la rotación, la traslación, etc.



Las activaciones siguen dependiendo de la ubicación, etc.

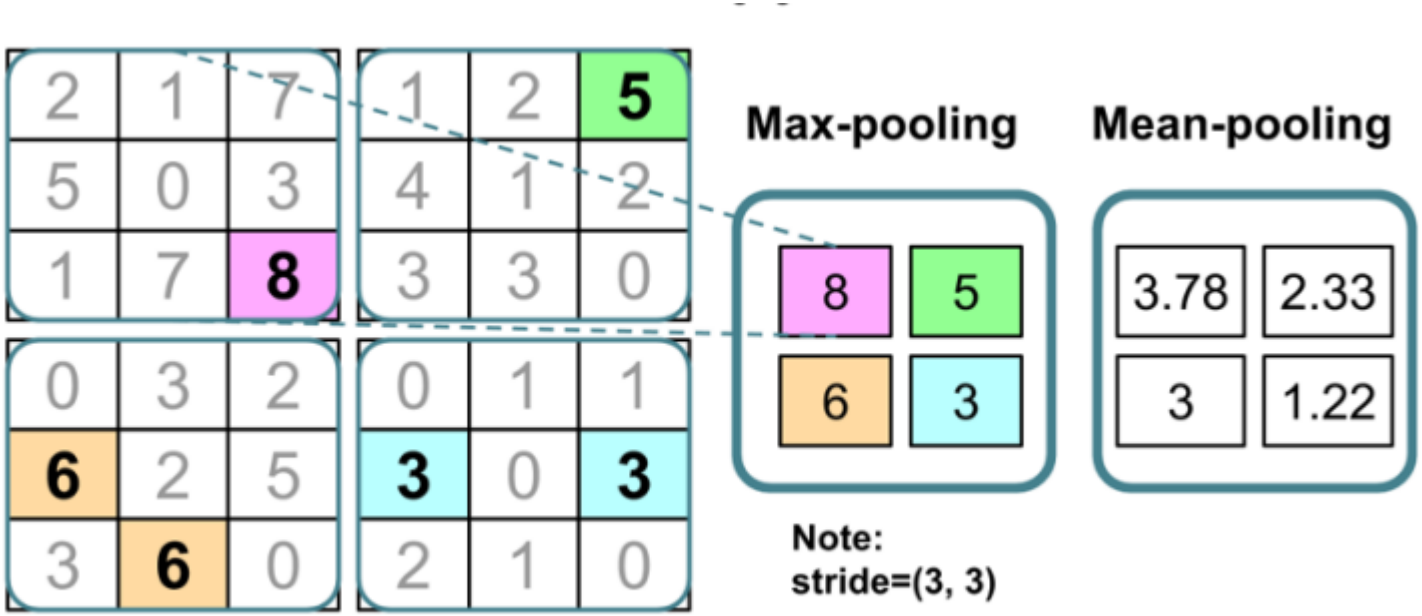
Pooling layers pueden ayudar con la invariancia Local



Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition.* Birmingham, UK: Packt Publishing, 2019. ISBN: 978-1789955750

Desventaja: Se pierde información.
Puede que no importe para la clasificación, pero si en aplicaciones en las que la posición relativa es importante (como el reconocimiento facial).
En la práctica para las CNN: todavía se recomienda algún preprocesamiento de imágenes

La Agrupación de Capas Puede Ayudar con la Invariancia Local



Note that typical pooling layers do not have any learnable parameters

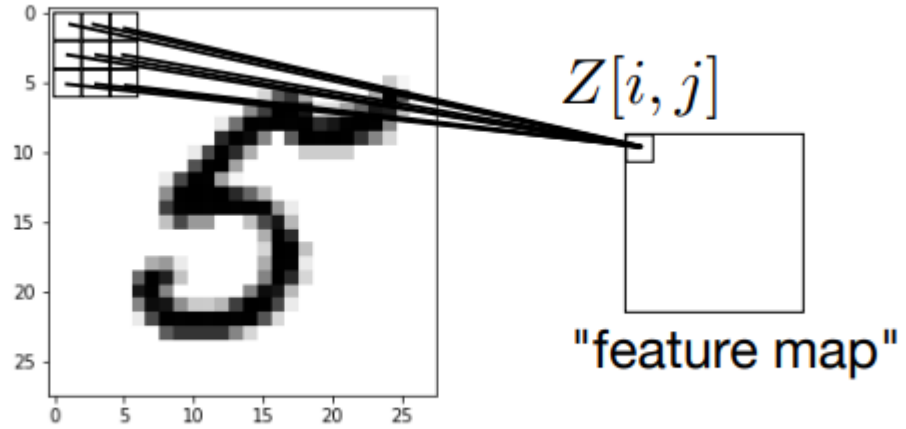
Una mirada más cercana a la capa convolucional

1. ¿Qué pueden hacer las CNNs?
2. Clasificación de imágenes
3. Conceptos básicos de CNNs
4. Filtros convolucionales y pesos compartidos
5. **Correlación cruzada vs convolución**
6. CNNs y backpropagation (retropropagación)
7. Arquitectura de las CNNs
8. Lo que pueden ver las CNN
9. CNNs en PyTorch

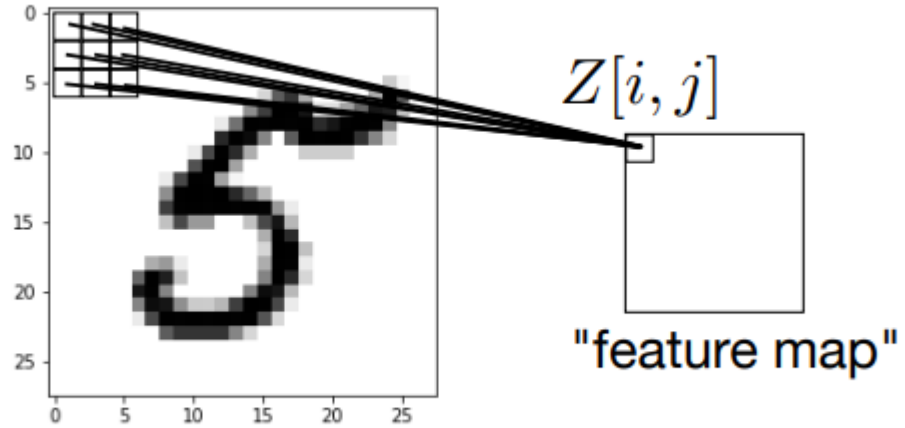
Correlación cruzada vs convolución

Jerga de aprendizaje profundo: la convolución en DL es en realidad correlación cruzada

La correlación cruzada es nuestro producto de puntos deslizantes sobre la imagen.



Correlación cruzada vs convolución



Correlación cruzada:

$$Z[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} K[u,v] A[i+u, j+v]$$

$$Z[i,j] = K \otimes A$$

Correlación cruzada vs convolución

Correlación cruzada

$$Z[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} K[u,v]A[i+u, j+v]$$

$$Z[i,j] = K \otimes A$$

La dirección del bucle se indica mediante los números rojos

1) -1,-1	2) -1,0	3) -1,1
4) 0,-1	5) 0,0	6) 0,1
7) 1,-1	8) 1,0	9) 1,1

Correlación cruzada vs convolución

Correlación cruzada

$$Z[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} K[u,v]A[i+u, j+v]$$

$$Z[i,j] = K \otimes A$$

Convolución

$$Z[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} K[u,v]A[i-u, j-v]$$

$$Z[i,j] = K * A$$

Básicamente, estamos rotando el kernel (o el campo receptivo) horizontal y verticalmente

9) -1,-1	8) -1,0	7) -1,1
6) 0,-1	5) 0,0	4) 0,1
3) 1,-1	2) 1,0	1) 1,1

[1]:
```
from scipy.signal import correlate2d, convolve2d
import torch
```

[2]:
```
a = torch.tensor([[1.1, 1.2, 1.3],
                  [2.1, 2.2, 2.3],
                  [3.1, 3.2, 3.3]])
```

[3]:
```
conv_pytorch = torch.nn.Conv2d(in_channels=1, out_channels=1, kernel_size=(3, 3))
with torch.no_grad():
    conv_pytorch.bias.zero_()

conv_pytorch.weight
```

[3]:
```
Parameter containing:
tensor([[[[ 0.1189, -0.1202,  0.2584],
          [-0.2276, -0.1327, -0.0296],
          [ 0.1741, -0.3225, -0.0288]]]], requires_grad=True)
```

[4]:
```
conv_weight_numpy = conv_pytorch.weight.detach().numpy().reshape(3, 3)
conv_weight_numpy
```

[4]:
```
array([[ 0.11893535, -0.12021737,  0.2583796 ],
       [-0.22761738, -0.13269596, -0.02961969],
       [ 0.17413345, -0.3224947 , -0.02875605]], dtype=float32)
```

Cross-correlation

[5]:
```
conv_pytorch(a.view(1, 1, 3, 3))
```

[5]:
```
tensor([[[[-1.1027]]]])
```

[6]:
```
correlate2d(a.numpy(), conv_weight_numpy, mode='valid')
```

[6]:
```
array([[-1.1026558]], dtype=float32)
```

[1]:
```
from scipy.signal import correlate2d, convolve2d
import torch
```

[2]:
```
a = torch.tensor([[1.1, 1.2, 1.3],
                  [2.1, 2.2, 2.3],
                  [3.1, 3.2, 3.3]])
```

[3]:
```
conv_pytorch = torch.nn.Conv2d(in_channels=1, out_channels=1, kernel_size=(3, 3))
with torch.no_grad():
    conv_pytorch.bias.zero_()

conv_pytorch.weight
```

[3]:
```
Parameter containing:
tensor([[[[ 0.1189, -0.1202,  0.2584],
          [-0.2276, -0.1327, -0.0296],
          [ 0.1741, -0.3225, -0.0288]]]], requires_grad=True)
```

[4]:
```
conv_weight_numpy = conv_pytorch.weight.detach().numpy().reshape(3, 3)
conv_weight_numpy
```

[4]:
```
array([[ 0.11893535, -0.12021737,  0.2583796 ],
       [-0.22761738, -0.13269596, -0.02961969],
       [ 0.17413345, -0.3224947 , -0.02875605]], dtype=float32)
```

Real convolution

[7]:
```
convolve2d(a.numpy(), conv_weight_numpy, mode='valid')
```

[7]:
```
array([[-0.2611365]], dtype=float32)
```

[8]:
```
a_mod = torch.tensor([[3.3, 3.2, 3.1],
                      [2.3, 2.2, 2.1],
                      [1.3, 1.2, 1.1]])


conv_pytorch(a_mod.view(1, 1, 3, 3))
```

[8]:
```
tensor([[[[-0.2611]]]])
```

Correlación cruzada vs convolución

Jerga de aprendizaje profundo: la convolución en DL es en realidad correlación cruzada

Convolución "real" tiene la propiedad asociativa:
$(A * B) * C = A * (B * C)$

En DL, generalmente no nos importa eso (a diferencia de muchas aplicaciones tradicionales de procesamiento de señales y visión por computadora).

Además, la correlación cruzada es más fácil de implementar.

Quizás el término "convolución" para la correlación cruzada se hizo popular, porque "Red neuronal correlacional cruzada" suena extraño

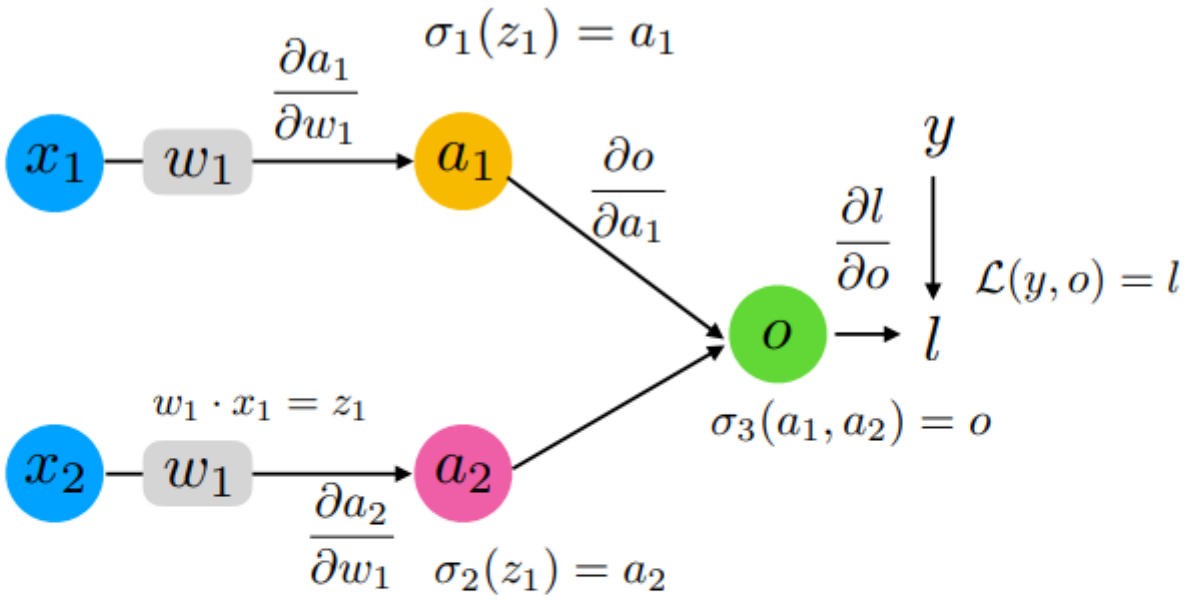
¿Cómo Funciona la Retropropagación en las CNNs?

1. ¿Qué pueden hacer las CNNs?
2. Clasificación de imágenes
3. Conceptos básicos de CNNs
4. Filtros convolucionales y pesos compartidos
5. Correlación cruzada vs convolución
6. **CNNs y backpropagation (retropropagación)**
7. Arquitectura de las CNNs
8. Lo que pueden ver las CNN
9. CNNs en PyTorch

Retropropagación en CNNs

El mismo concepto general que antes: regla de cadena multivariable, pero ahora con una restricción de peso compartido adicional

¿Recuerdan la lección 6? Gráfico con Pesos Compartidos



Upper path

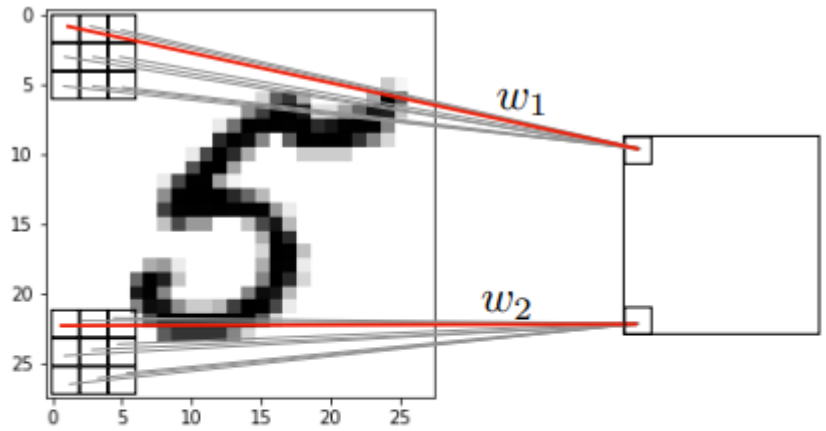
$$\frac{\partial l}{\partial w_1} = \frac{\partial l}{\partial o} \cdot \frac{\partial o}{\partial a_1} \cdot \frac{\partial a_1}{\partial w_1} + \frac{\partial l}{\partial o} \cdot \frac{\partial o}{\partial a_2} \cdot \frac{\partial a_2}{\partial w_1}$$ (multivariable chain rule)

Lower path

Retropropagación en CNNs

El mismo concepto general que antes: regla de cadena multivariable, pero ahora con una restricción de peso compartido adicional

Debido al peso compartido: w1 = w2



Optional averaging

weight update:

$$w_1 := w_2 := w_1 - \eta \cdot \frac{1}{2}\left(\frac{\partial \mathcal{L}}{\partial w_1} + \frac{\partial \mathcal{L}}{\partial w_2}\right)$$

¿Cuáles son algunas de las arquitecturas de CNNs más Comunes?

1. ¿Qué pueden hacer las CNNs?
2. Clasificación de imágenes
3. Conceptos básicos de CNNs
4. Filtros convolucionales y pesos compartidos
5. Correlación cruzada vs convolución
6. CNNs y backpropagation (retropropagación)
7. **Arquitectura de las CNNs**
8. Lo que pueden ver las CNN
9. CNNs en PyTorch

Breakthrough para las CNNs: AlexNet e ImageNet

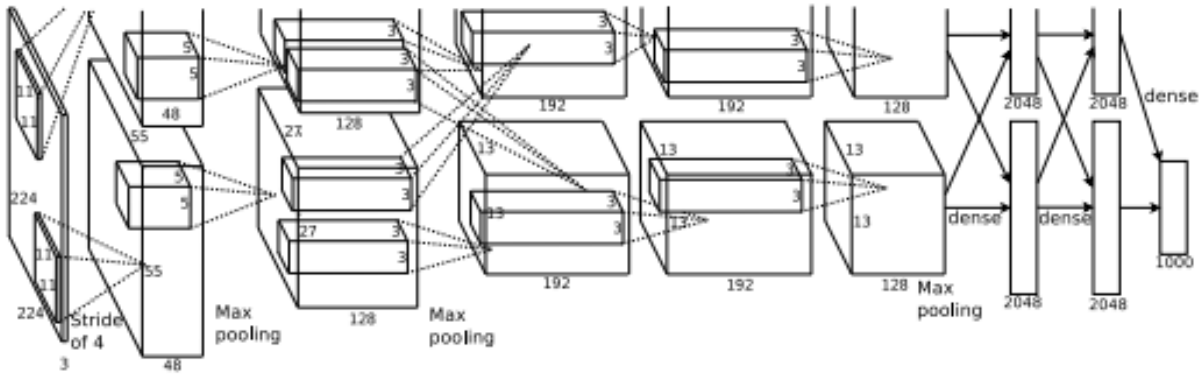


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input 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.

AlexNet logró un error del 15.4% en el top-5 en 2012, el segundo mejor ni siquiera estuvo cerca: 26.2% (hoy en día ~3% error en ImageNet)

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

Breakthrough para las CNNs: AlexNet e ImageNet

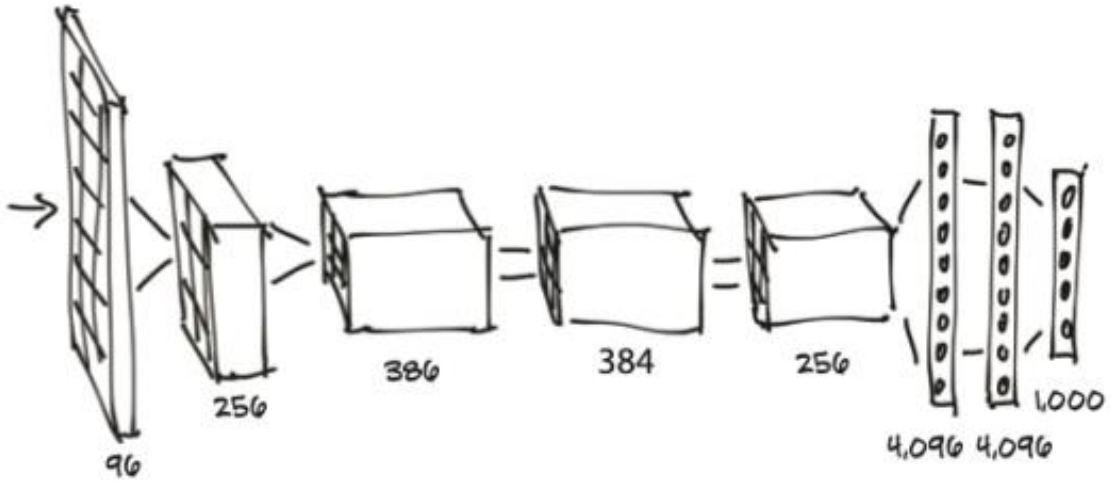


Figure 2.3 The AlexNet architecture

Stevens, Eli, Luca Antiga, and Thomas Viehmann. Deep learning with PyTorch. Manning Publications, 2020

Breakthrough para las CNNs: AlexNet e ImageNet

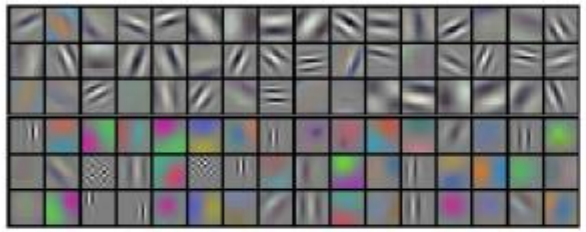


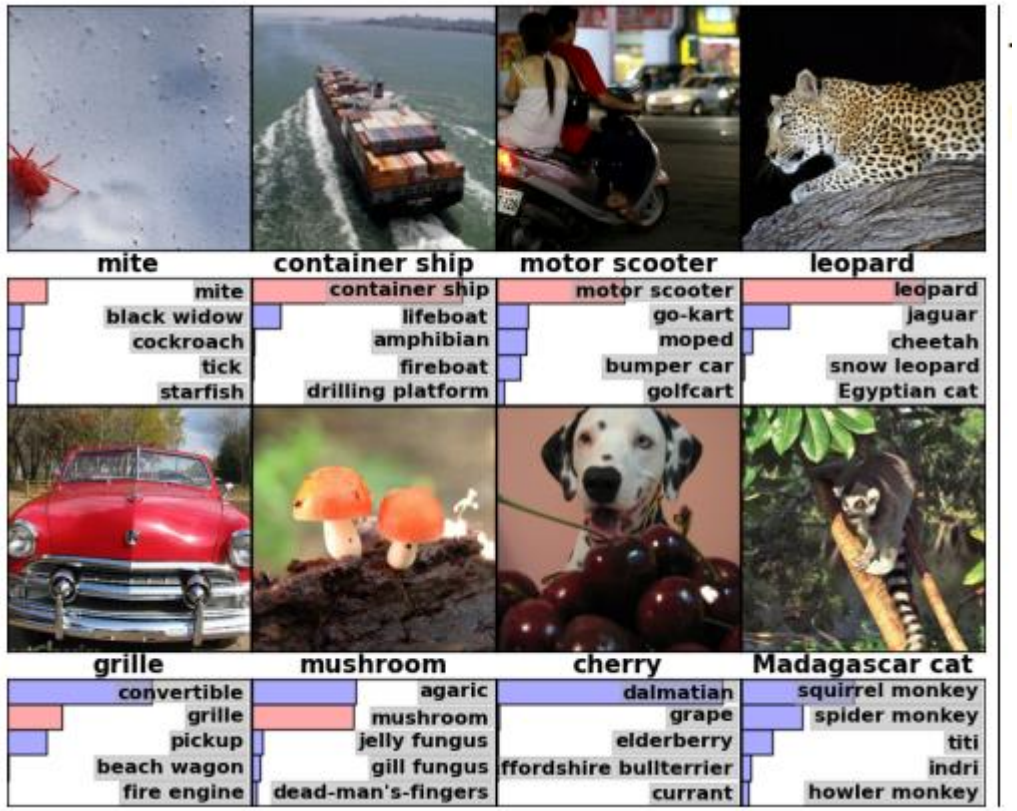
Figure 3: 96 convolutional kernels of size $11 \times 11 \times 3$ learned by the first convolutional layer on the $224 \times 224 \times 3$ input images. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2. See Section 6.1 for details.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248–255).

Figure 4: **(Left)** Eight ILSVRC-2010 test images and the five labels considered most probable by our model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5). **(Right)** Five ILSVRC-2010 test images in the first column. The remaining columns show the six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

Breakthrough para las CNNs: AlexNet e ImageNet

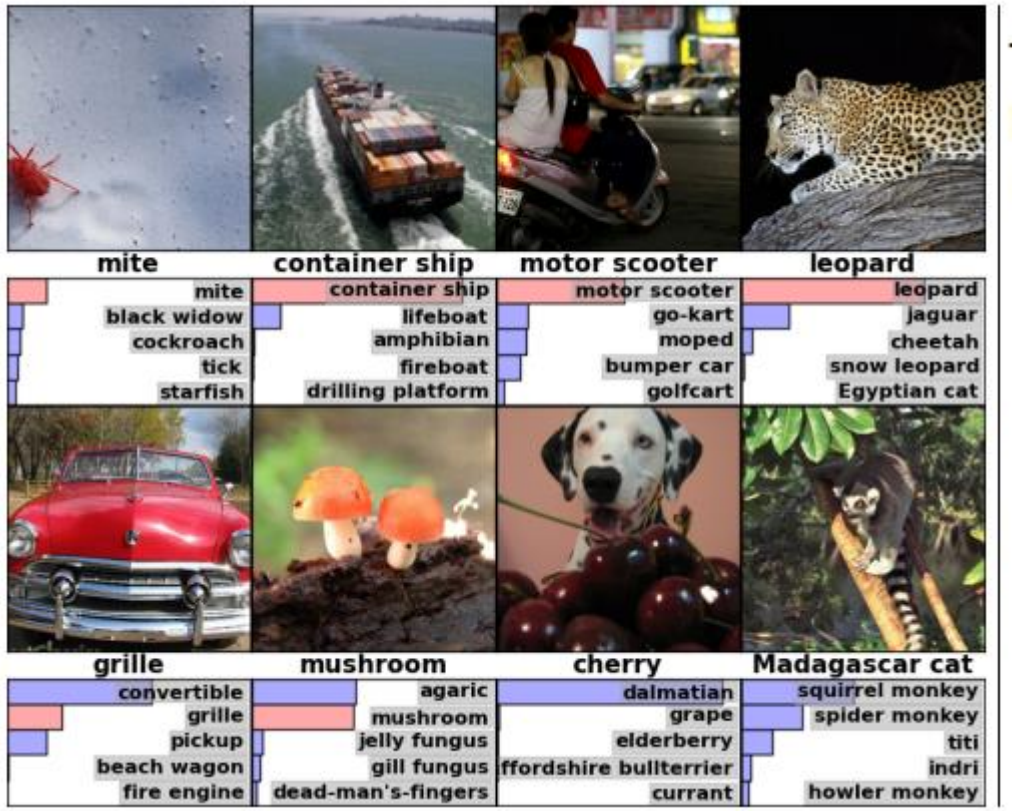


El conjunto de ImageNet que se utilizó tiene ~ 1,2 millones de imágenes y 1000 clases

La precisión se mide como el rendimiento del top-5: Predicción correcta si la etiqueta verdadera coincide con una de las predicciones del modelo del top-5.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

Breakthrough para las CNNs: AlexNet e ImageNet



Tenga en cuenta que las entradas de red reales seguían siendo imágenes de 224x224 (recortes aleatorios de imágenes de 256x256)

224x224 sigue siendo un tamaño bueno / razonable en la actualidad (224 * 224 * 3 = 150,528 características)

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

Arquitecturas comunes de CNN

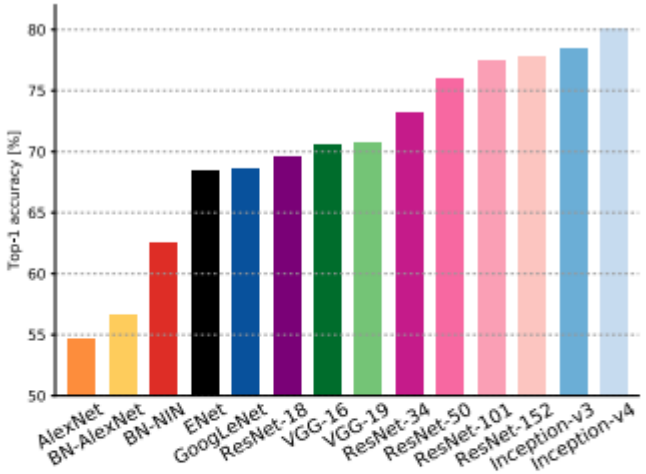


Figure 1: **Top1 *vs.* network.** Single-crop top-1 validation accuracies for top scoring single-model architectures. We introduce with this chart our choice of colour scheme, which will be used throughout this publication to distinguish effectively different architectures and their correspondent authors. Notice that networks of the same group share the same hue, for example ResNet are all variations of pink.

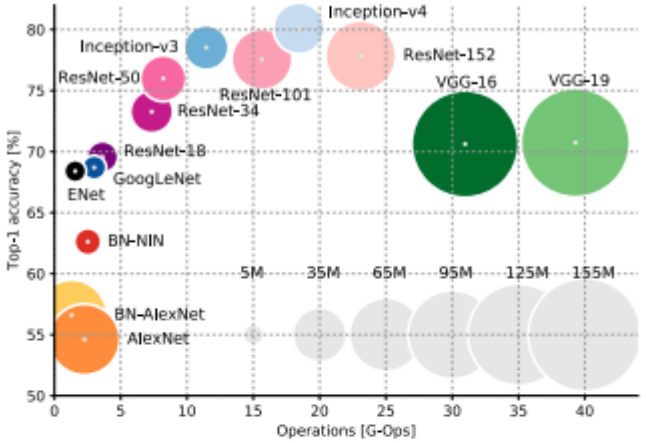
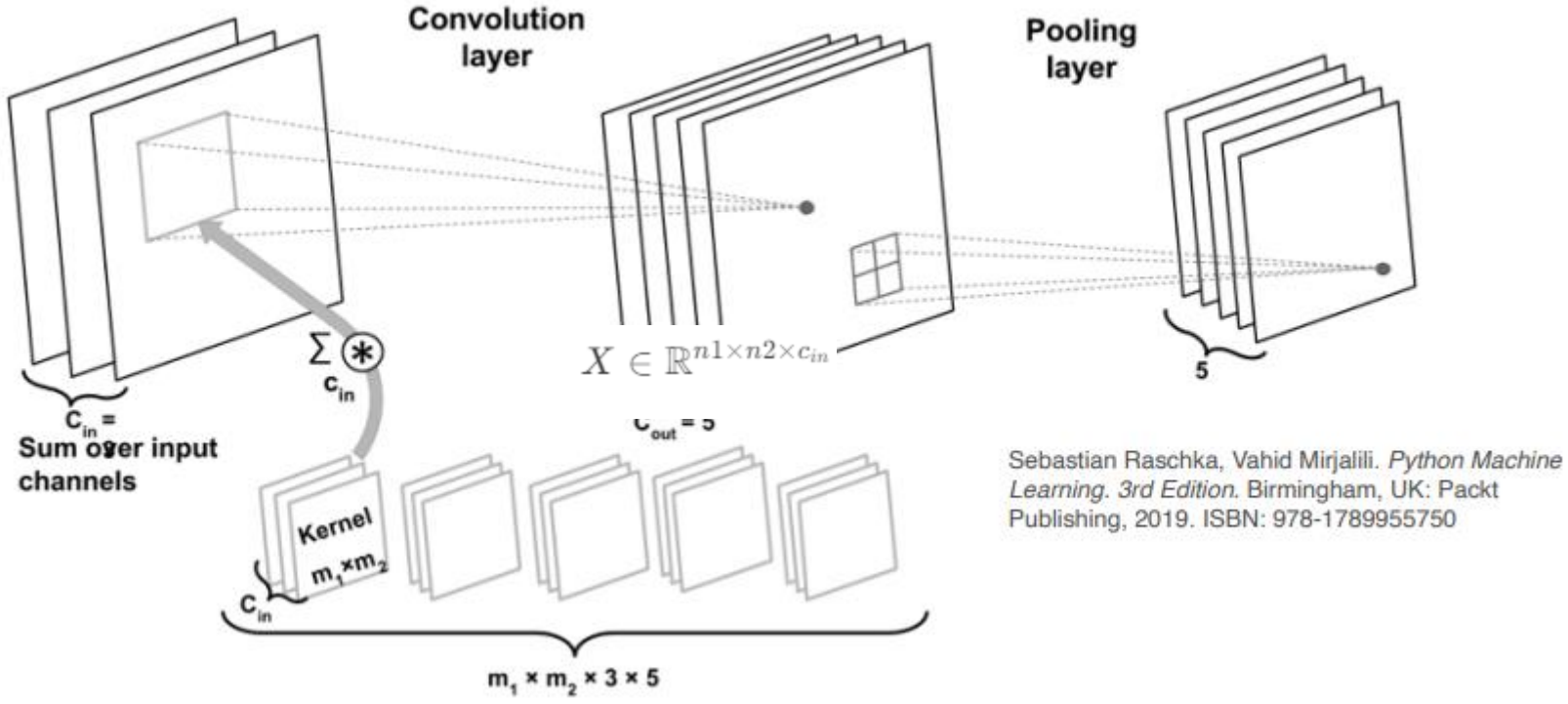


Figure 2: **Top1 *vs.* operations, size $\propto$ parameters.** Top-1 one-crop accuracy versus amount of operations required for a single forward pass. The size of the blobs is proportional to the number of network parameters; a legend is reported in the bottom right corner, spanning from 5×10^6 to 155×10^6 params. Both these figures share the same y-axis, and the grey dots highlight the centre of the blobs.

Canziani, A., Paszke, A., & Culurciello, E. (2016). An analysis of deep neural network models for practical applications. arXiv preprint arXiv:1605.07678

Convoluciones con múltiples canales



Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition.* Birmingham, UK: Packt Publishing, 2019. ISBN: 978-1789955750

Dimensión de la imagen: $X \in \mathbb{R}^{n1 \times n2 \times c_{in}}$ en formato NWHC,
CUDA y PyTorch usan NCWH

Interpretación de las capas de ConvNet (ocultas)

1. ¿Qué pueden hacer las CNNs?
2. Clasificación de imágenes
3. Conceptos básicos de CNNs
4. Filtros convolucionales y pesos compartidos
5. Correlación cruzada vs convolución
6. CNNs y backpropagation (retropropagación)
7. Arquitectura de las CNNs
8. **Lo que pueden ver las CNN**
9. CNNs en PyTorch

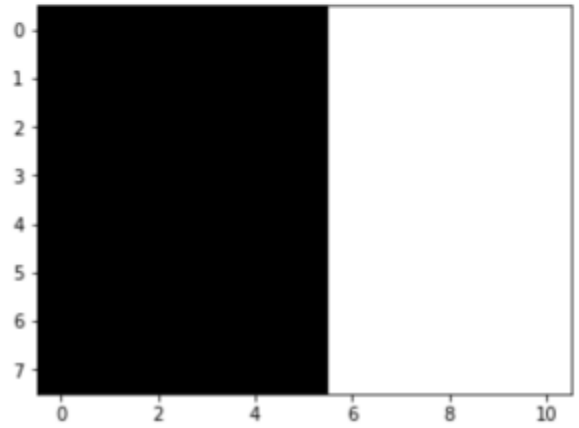
Lo que pueden ver las CNN

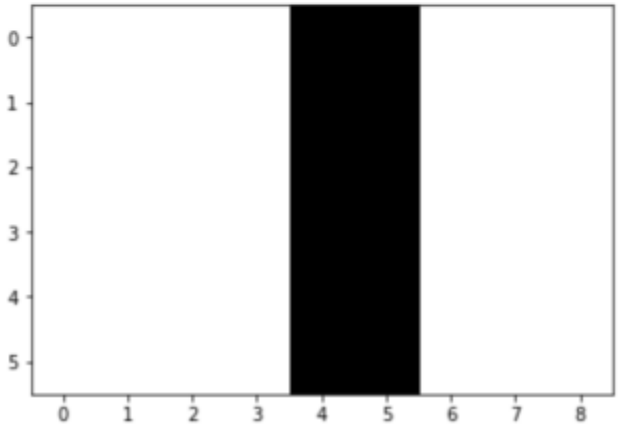
Ejemplo simple: detector de borde vertical

```
conv.weight[0, 0, :, :] = torch.tensor([[1, 0, -1],
                                        [1, 0, -1],
                                        [1, 0, -1]]).float()

t = torch.tensor([
[0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1.],
[0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1.],
[0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1.],
[0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1.],
[0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1.],
[0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1.],
[0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1.],
[0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1.],
])

plt.imshow(t, cmap='gray');
```

(De la investigación clásica en visión por computadora)

```
tt = torch.zeros([1, 1] + list(t.size()))
tt[0, 0, :, :] = t
after = conv(tt)
plt.imshow(after[0, 0, :, :].detach().numpy(), cmap='gray');
```





Lo que pueden ver las CNN

Ejemplo simple: detector de borde vertical

```
conv = torch.nn.Conv2d(in_channels=1,
                       out_channels=1,
                       kernel_size=(3, 3))
```

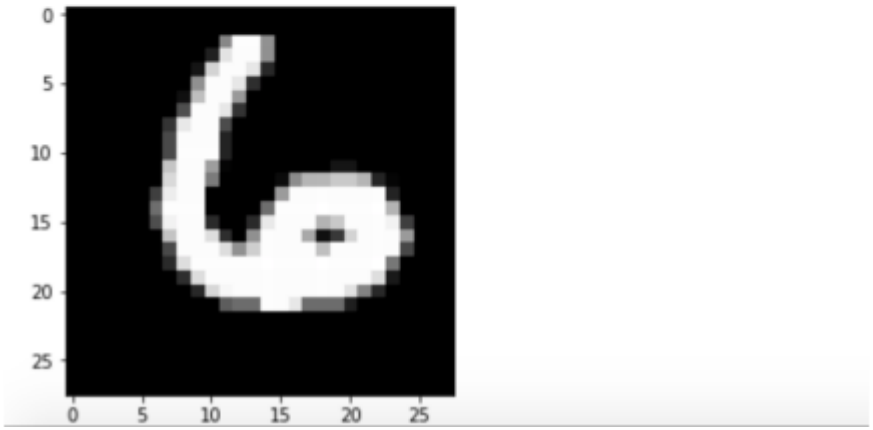
```
conv.weight.size()
```

```
torch.Size([1, 1, 3, 3])
```

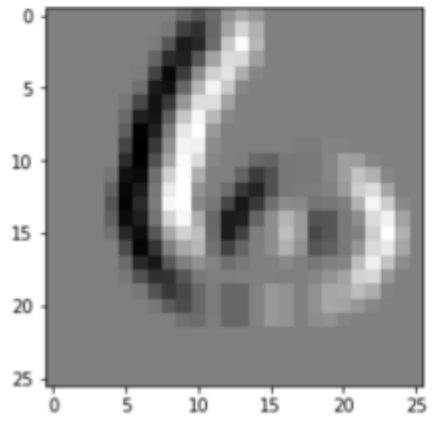
```
conv.weight[0, 0, :, :] = torch.tensor([[1, 0, -1],
                                        [1, 0, -1],
                                        [1, 0, -1]]).float()
conv.bias[0] = torch.tensor([0.]).float()
```

```
images_after = conv(images)
```

```
plt.imshow(images[5, 0], cmap='gray');
```




```
plt.imshow(images_after[5, 0].detach().numpy() , cmap='gray');
```



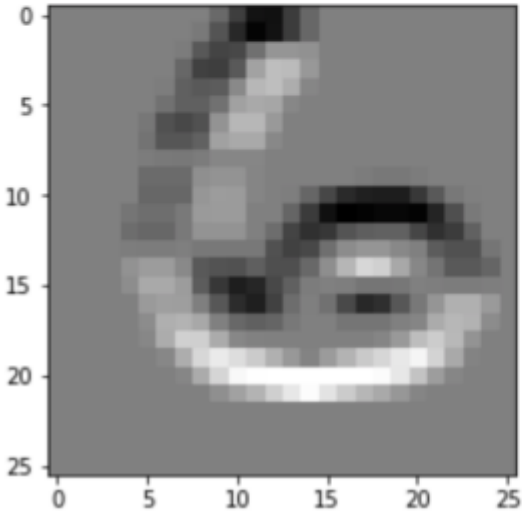
Lo que pueden ver las CNN

Ejemplo simple: detector de borde horizontal

```
conv.weight[0, 0, :, :] = torch.tensor([[1, 1, 1],
                                        [0, 0, 0],
                                        [-1, -1, -1]]).float()
conv.bias[0] = torch.tensor([0.]).float()
```

```
images_after2 = conv(images)
```

```
plt.imshow(images_after2[5, 0].detach().numpy() , cmap='gray');
```



Una CNN aprende la forma de los kernels basándose en la optimización de la función de costo (por ejemplo, minimizando una pérdida particular para lograr una buena precisión de clasificación)

Lo que pueden ver las CNN

¿Qué patrones del conjunto de entrenamiento activan el mapa de características?

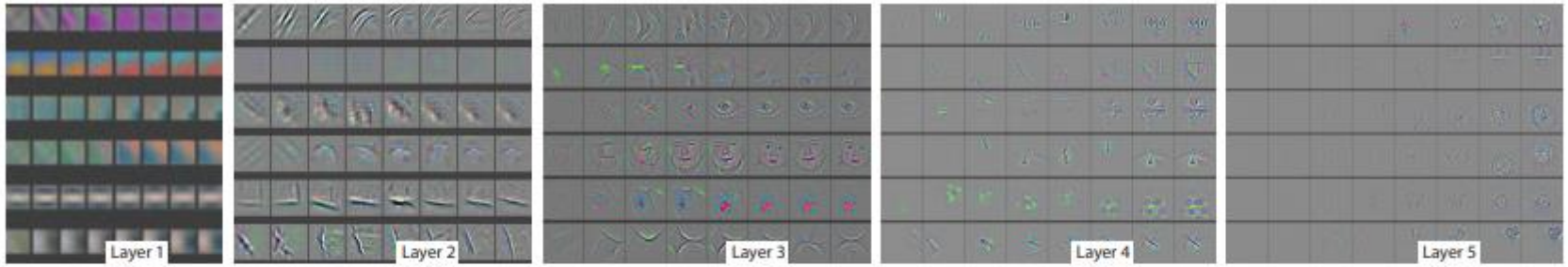


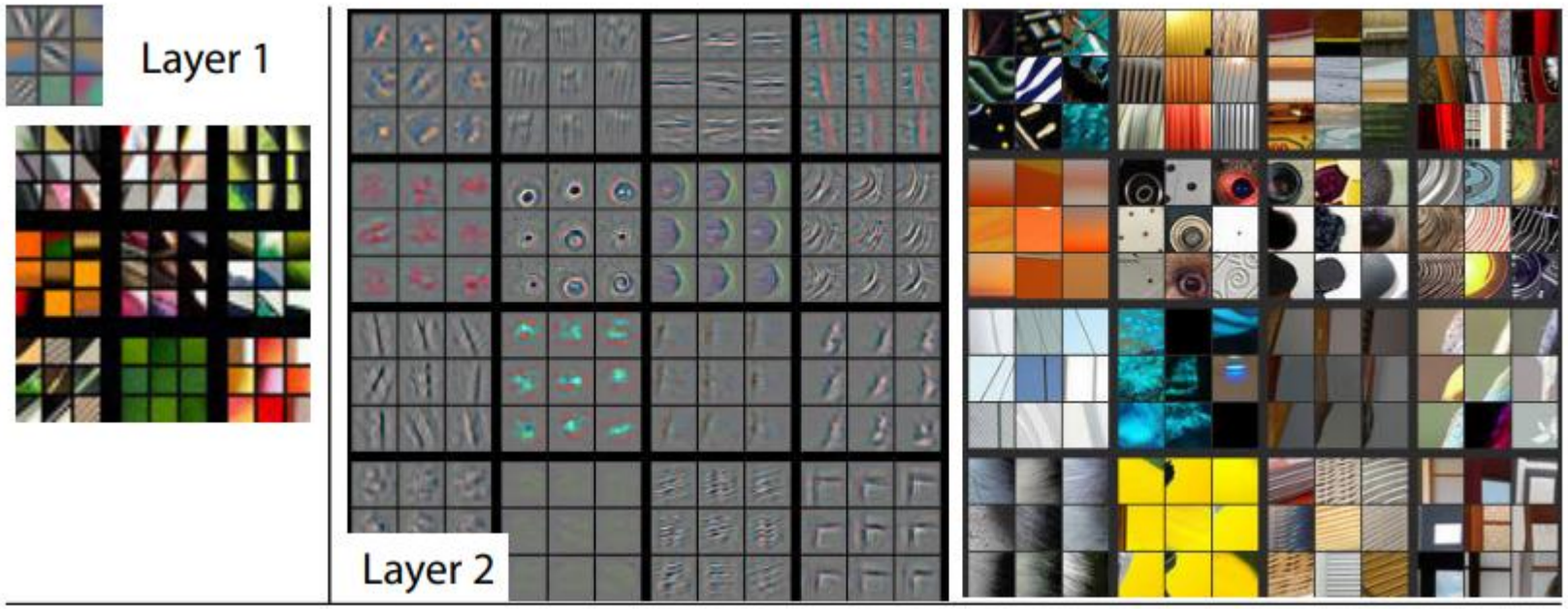
Fig. 4. Evolution of a randomly chosen subset of model features through training. Each layer's features are displayed in a different block. Within each block, we show a randomly chosen subset of features at epochs [1,2,5,10,20,30,40,64]. The visualization shows the strongest activation (across all training examples) for a given feature map, projected down to pixel space using our deconvnet approach. Color contrast is artificially enhanced and the figure is best viewed in electronic form.

Zeiler, M. D., & Fergus, R. (2014, September). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818-833). Springer, Cham.

Método: retropropagar señales de activación fuertes en capas ocultas a las imágenes de entrada, luego aplicar "deshacer" para asignar los valores al espacio de píxeles original para la visualización

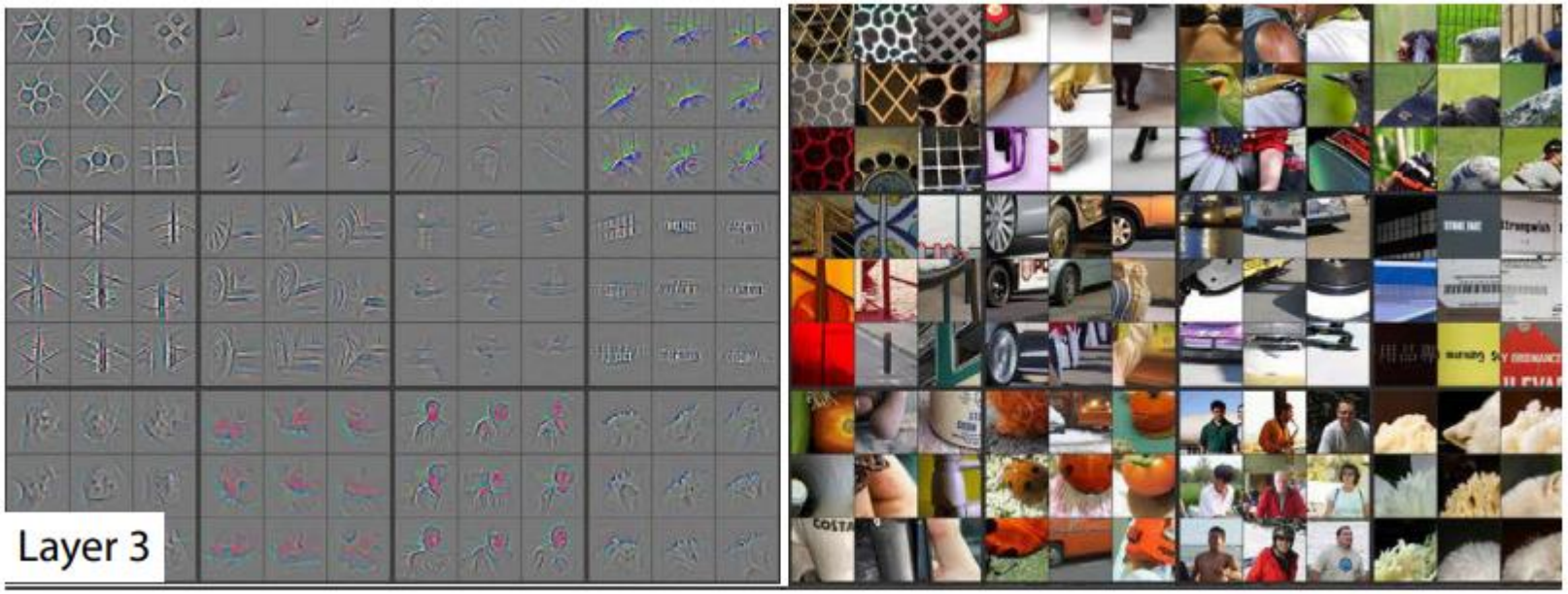
Lo que pueden ver las CNN

¿Qué patrones del conjunto de entrenamiento activan el mapa de características?



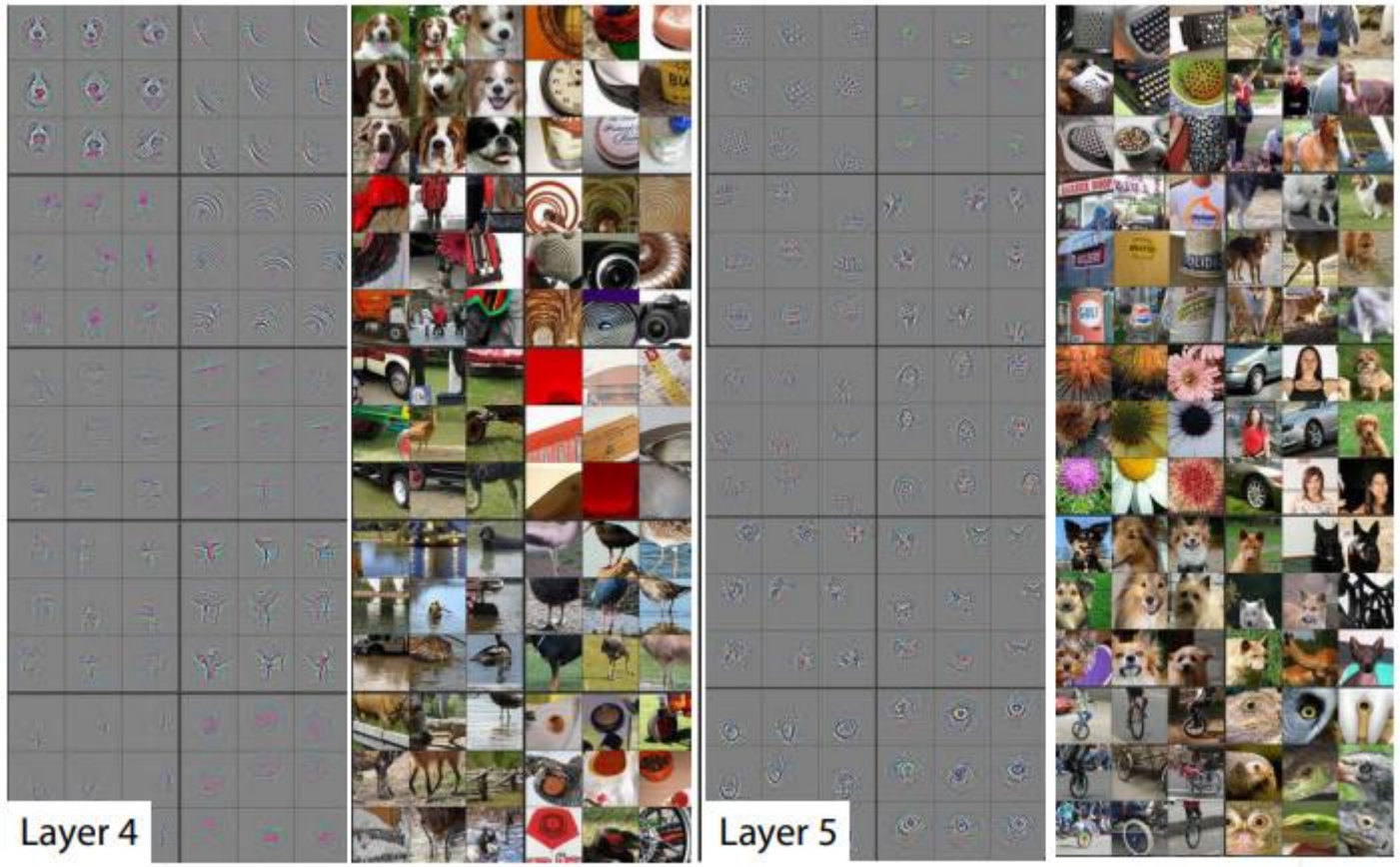
Lo que pueden ver las CNN

¿Qué patrones del conjunto de entrenamiento activan el mapa de características?



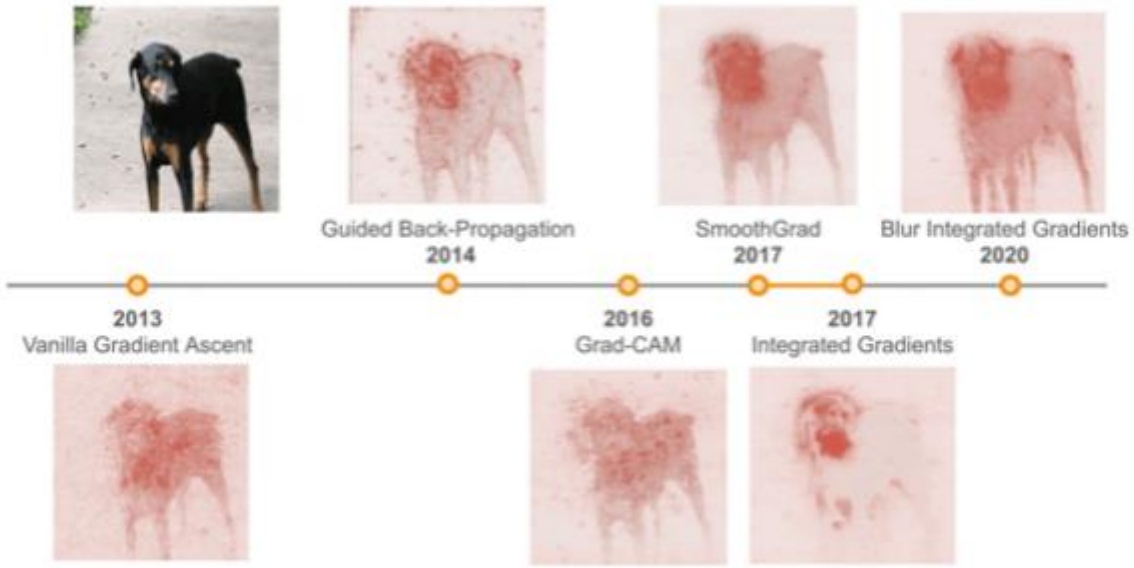
Lo que pueden ver las CNN

¿Qué patrones del conjunto de entrenamiento activan el mapa de características?



Lo que pueden ver las CNN

¿Qué patrones del conjunto de entrenamiento activan el mapa de características?



https://thegradient.pub/a-visual-history-of-interpretation-for-image-recognition/

Interpretación de las capas de ConvNet (ocultas)

1. ¿Qué pueden hacer las CNNs?
2. Clasificación de imágenes
3. Conceptos básicos de CNNs
4. Filtros convolucionales y pesos compartidos
5. Correlación cruzada vs convolución
6. CNNs y backpropagation (retropropagación)
7. Arquitectura de las CNNs
8. Lo que pueden ver las CNN
9. **CNNs en PyTorch y Keras**