

Introducción al Aprendizaje Profundo

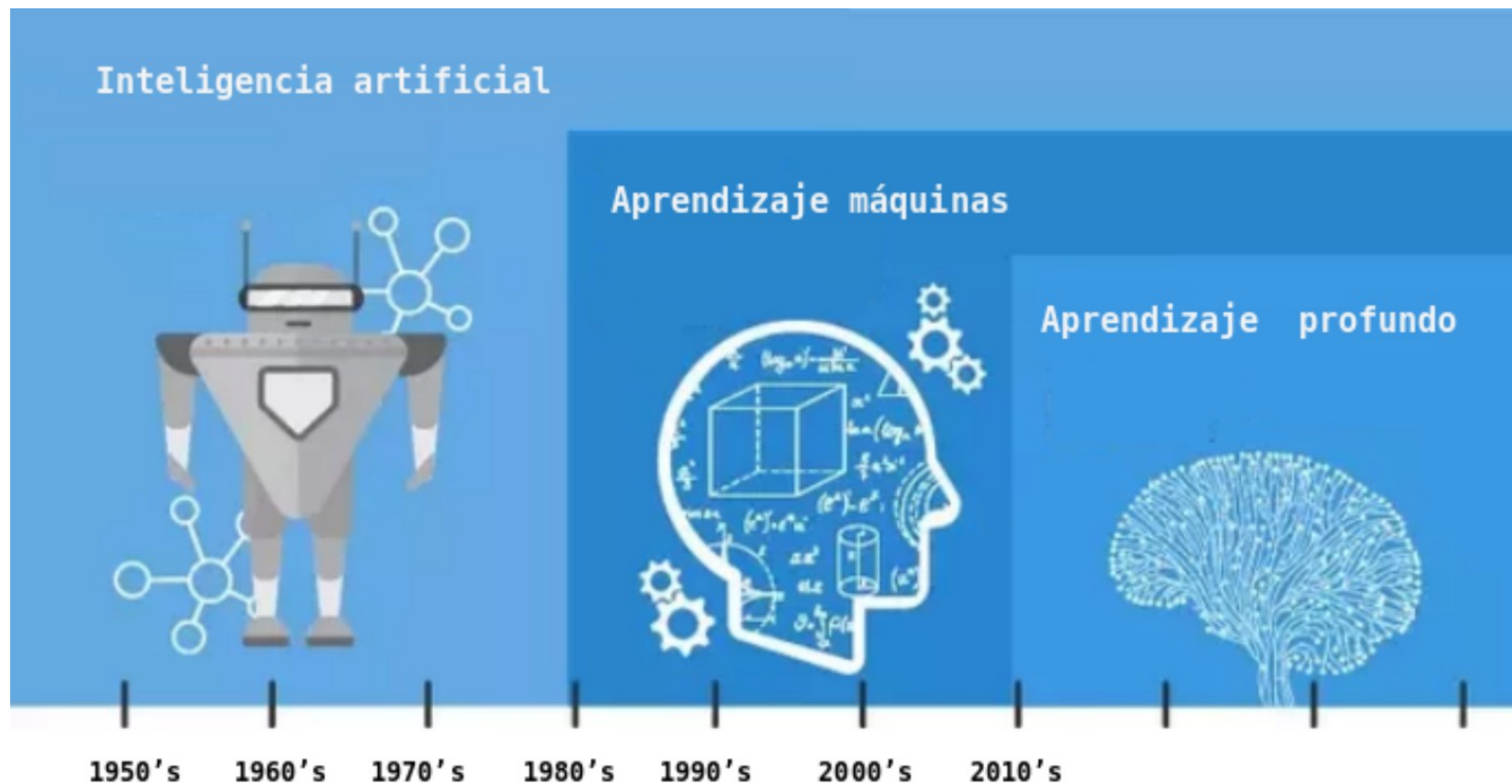
Bere & Ricardo Montalvo Lezama

github.com/richardtml/riiaa-20-aa

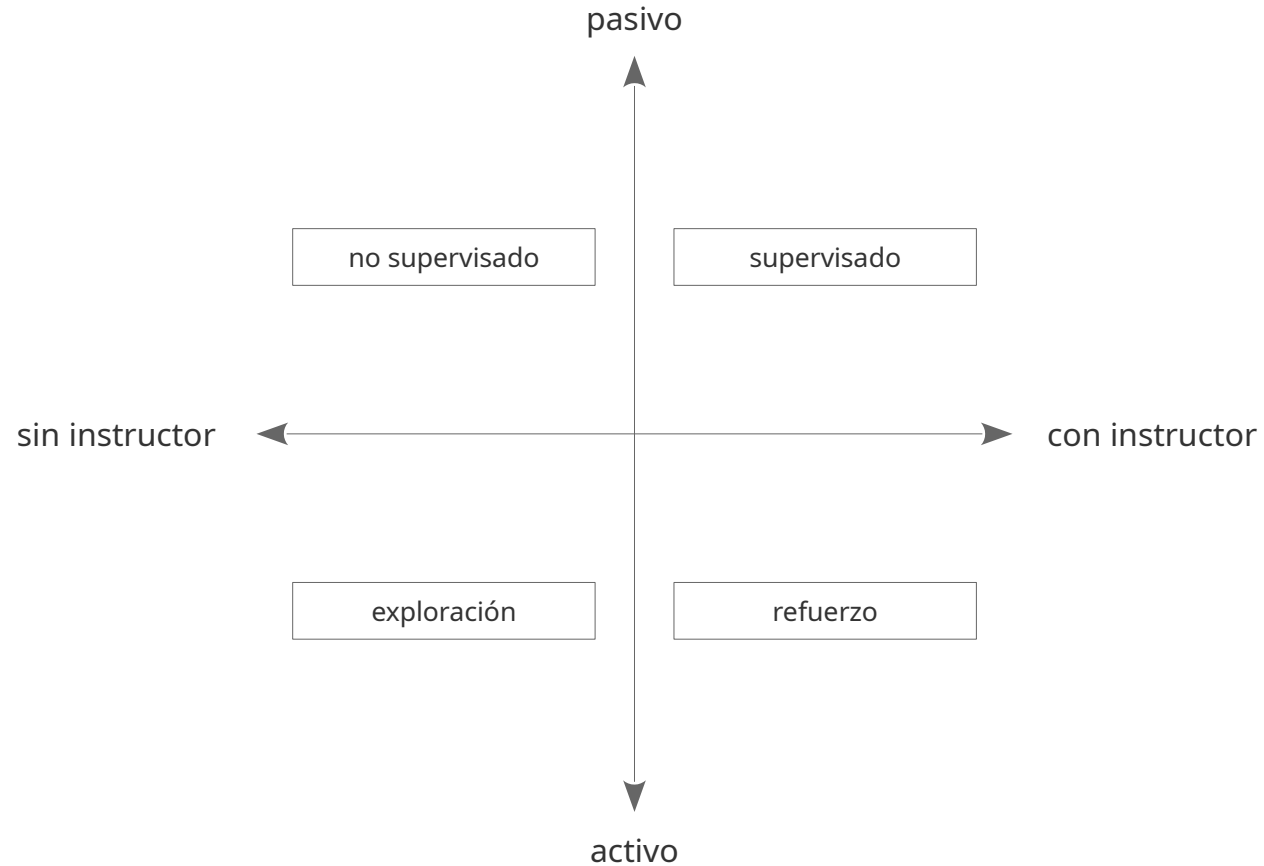


Agosto 2020

IA, AM y AP



Tipos de aprendizaje



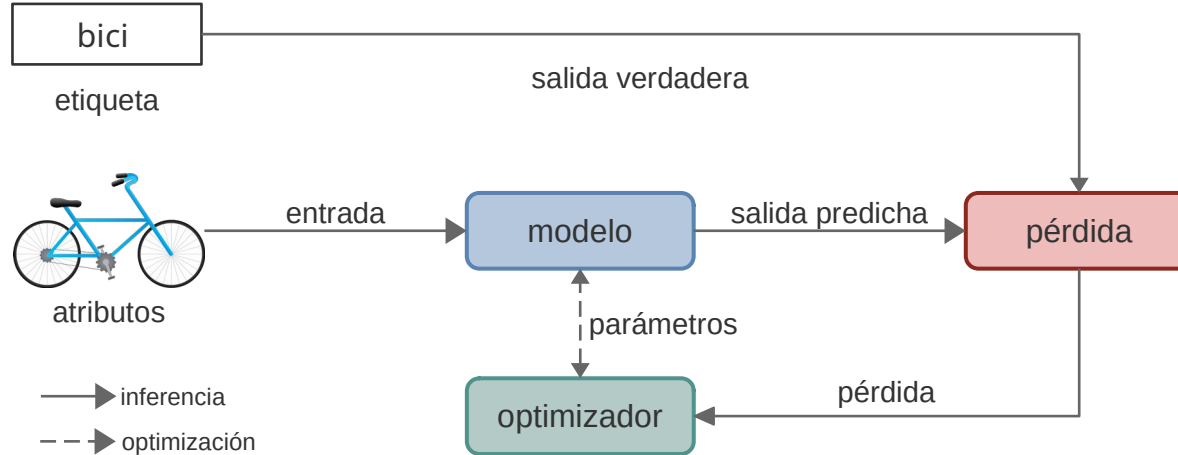
¿Cómo funciona el aprendizaje de máquinas?

- Programas que aprenden a partir de ejemplos.
- Se aprende un modelo: función parametrizada.



Aprendizaje supervisado

- La función de pérdida compara la salida verdadera con la salida predicha.
- Se minimiza la pérdida para actualizar los parámetros del modelo.
- El más común de los tipos de aprendizaje.



Una receta con mucho éxito

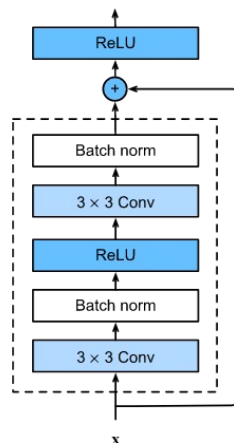
IMAGENET

14M de imágenes
21K categorías

YouTube-8M

8M de videos
4.8K categorías

+



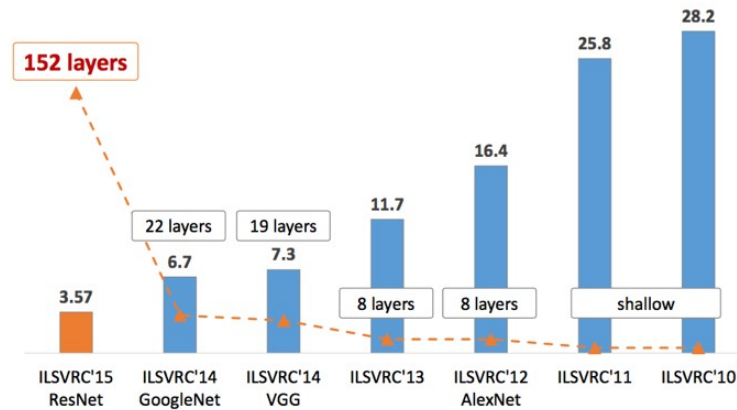
+



PyTorch

TensorFlow

=



datos masivos
etiquetados

arquitecturas
sofisticadas

infraestructura

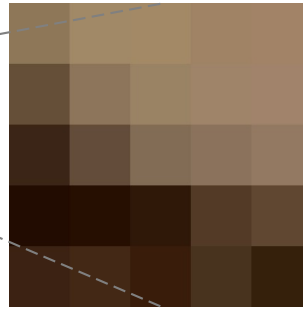
estado
del arte

Representación de imágenes

- Se representan con una matriz de valores de píxeles por cada color.



3x224x224



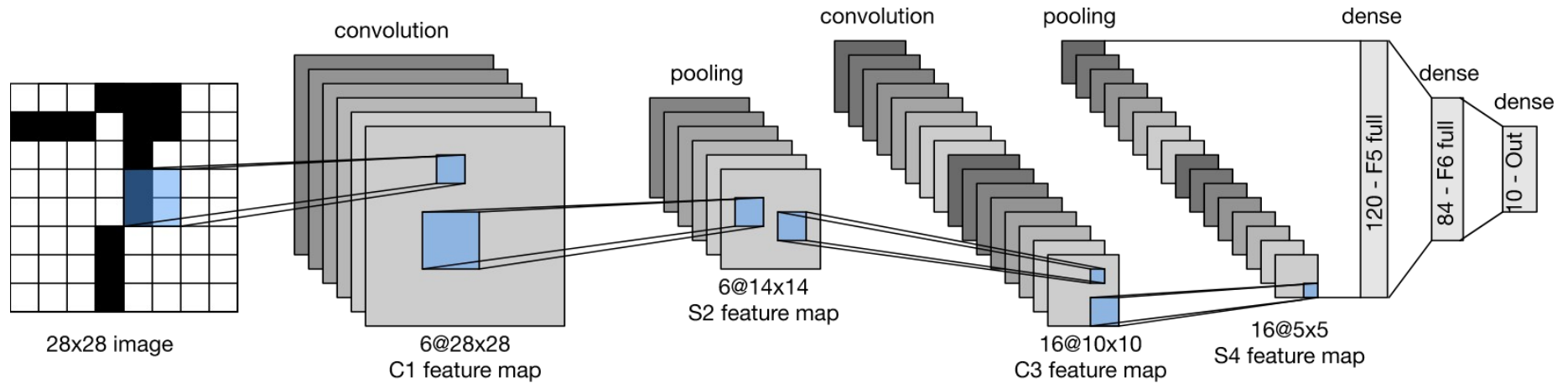
3x5x5

						33	4	48	9	61
					42	2	39	5	28	52
										29
13	4	27	11	5					2	13
3	29	1	38	48					6	25
12	8	31	4	29					36	
67	23	4	33	15					58	
36	75	12	8	42						

canales RGB

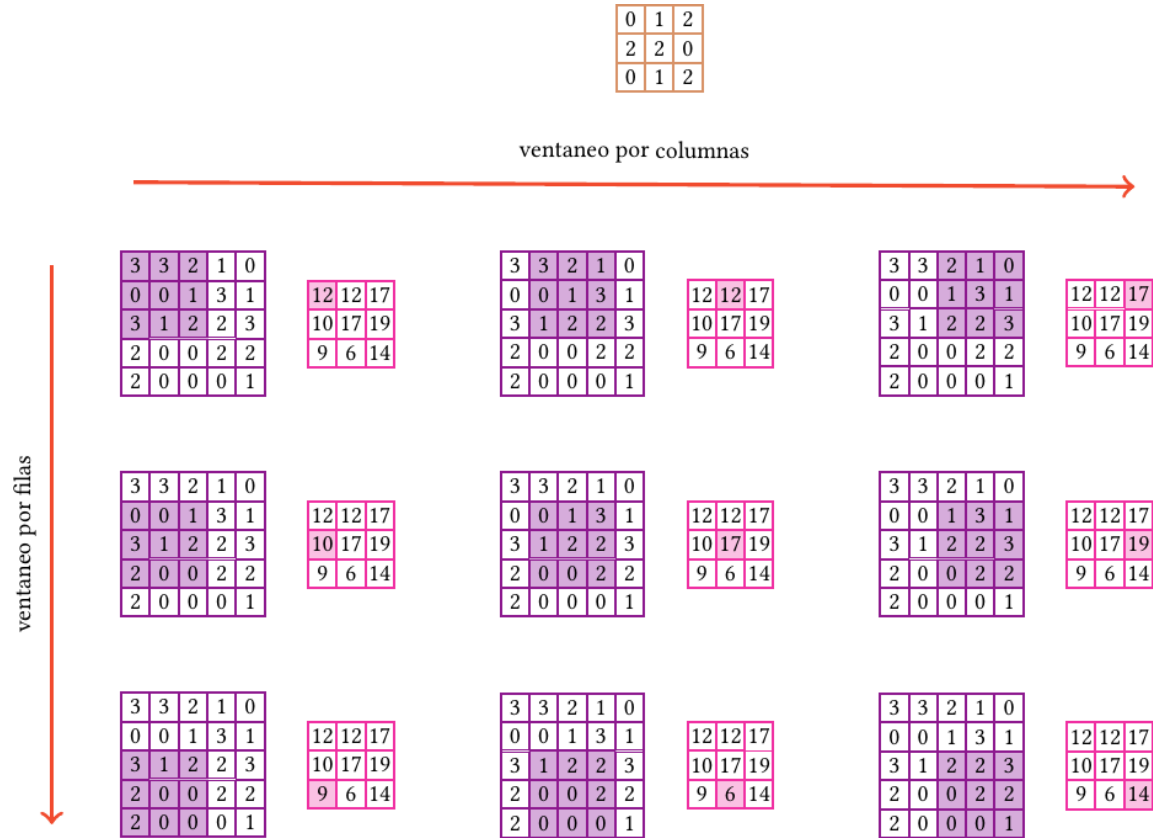
- MNIST 1x28x28, CIFAR-10 3x32x32, ImageNet 3x256x256.

Redes Convolucionales



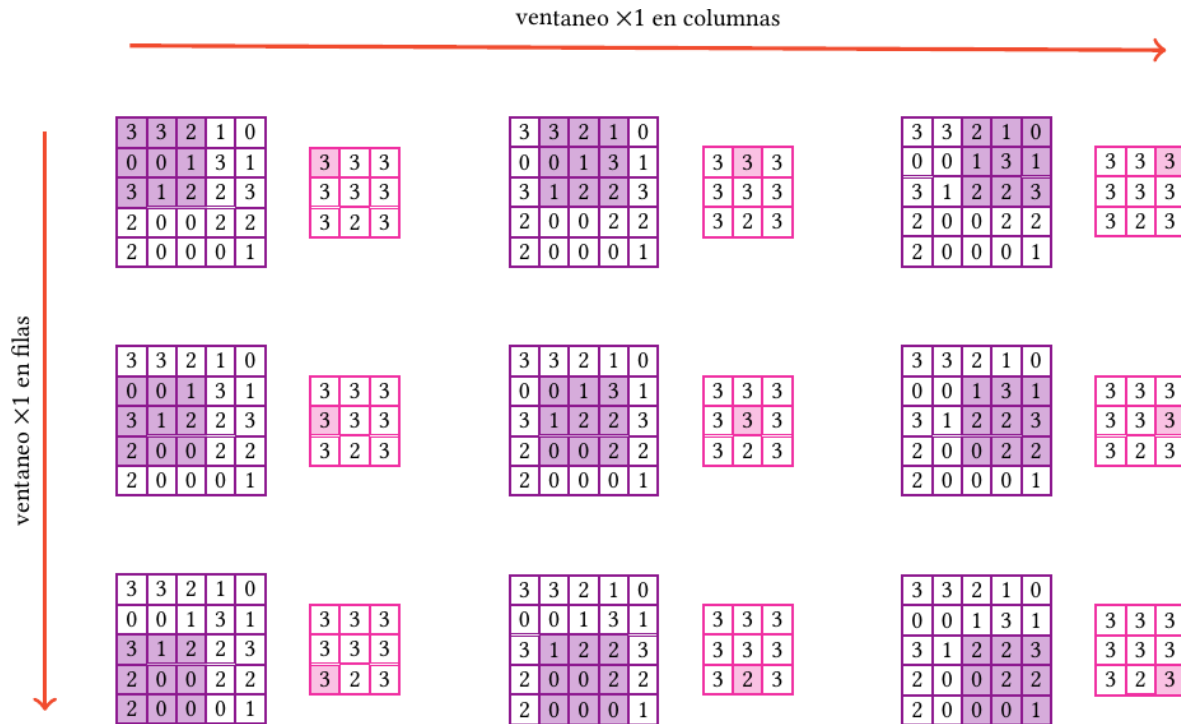
Arquitectura LeNet

Convolución



Convolución: entrada 5×5 , salida 3×3 , filtro 3×3 .

Muestreo



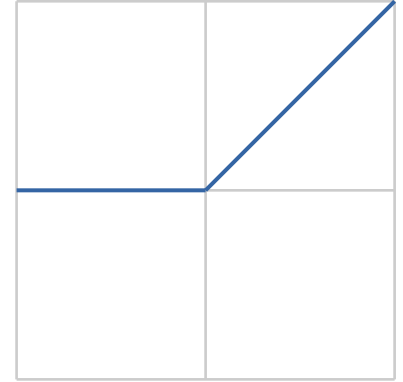
Muestreo máximo: entrada 5×5 , salida 3×3 , paso 1×1 .

Funciones de activación (I)

- Función no lineal a la salida de la neurona.
 - ReLU: capas intermedias.
 - Sigmoide: clasificación binaria.
 - Softmax: clasificación multiclase.

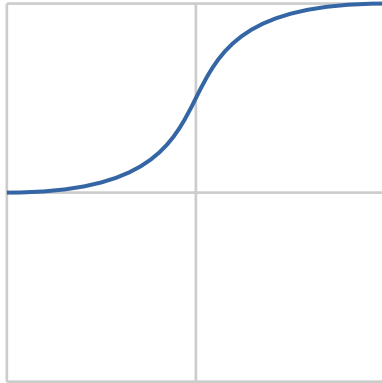
ReLU

$$\text{relu}(x) = \max(0, x)$$



Sigmoide

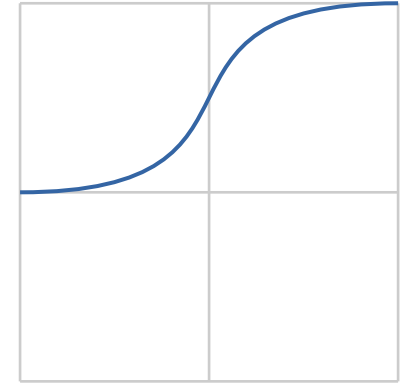
$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)}$$



Softmax

$$z = \text{softmax}(x)$$

$$z_j = \frac{\exp(x_j)}{\sum_k \exp(x_k)}$$



Normalización por lote

- Control de la media y la varianza en etapas de la red.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\dots m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

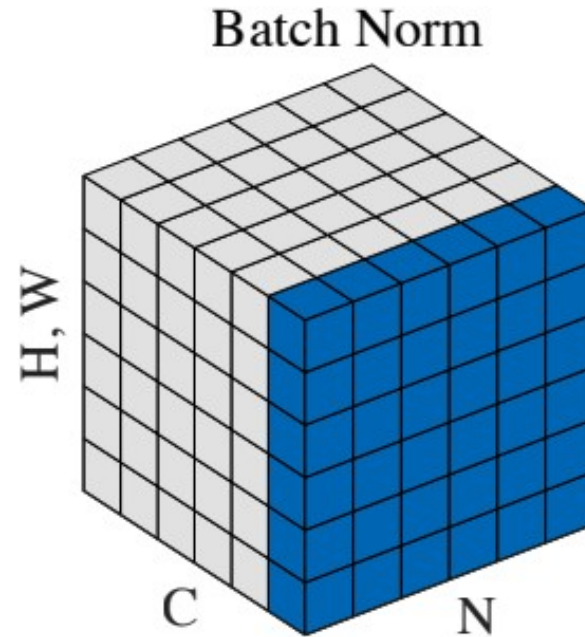
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

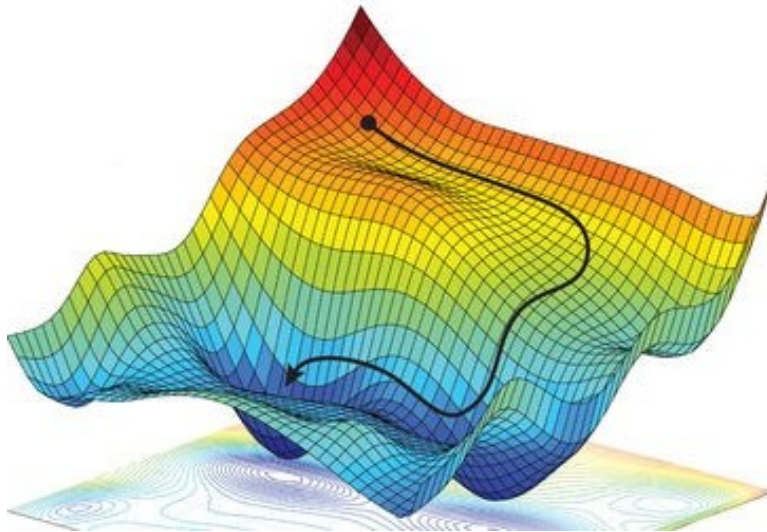
$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.



Descenso por gradiente

- Avanzar hacia la dirección con menor pérdida.



repetir hasta converger:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$



¡tiempo de programar!
1a_cnn.ipynb

Tareas de Visión

clasificación



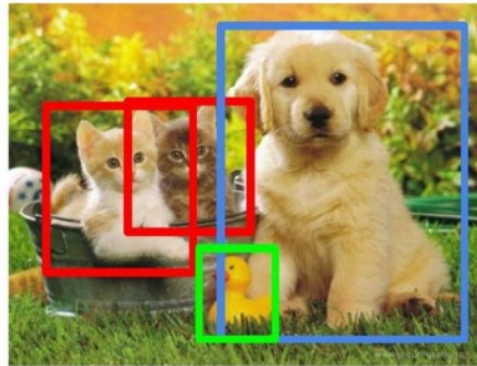
gato

clasificación
+ localización



gato

detección



gato, pato, pato

segmentación



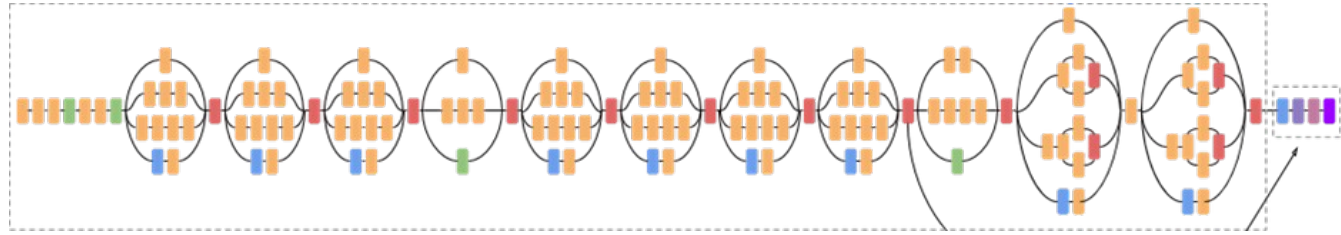
gato, pato, pato

un objeto

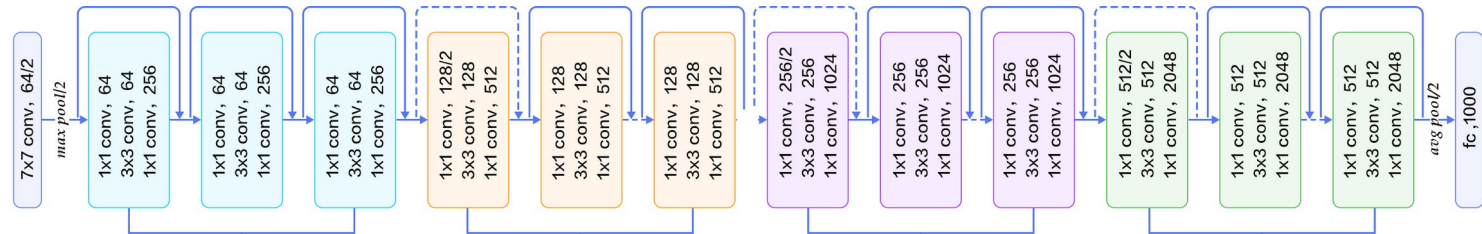
múltiples objetos

Arquitecturas

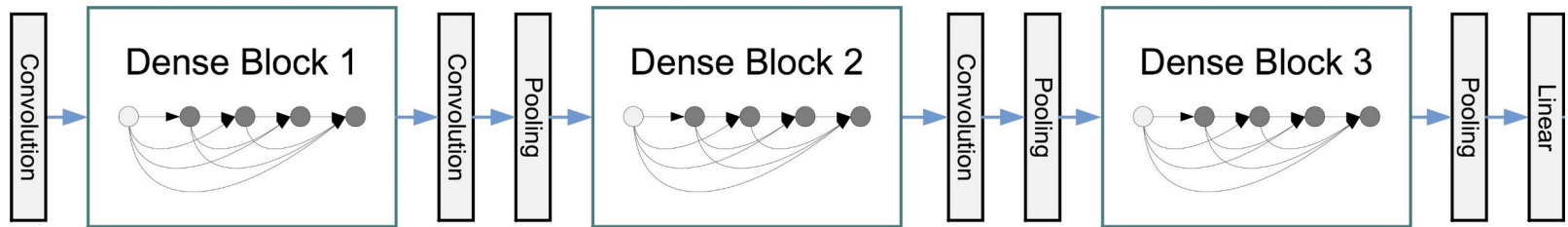
Inception



ResNet



DenseNet





¡Gracias!

Ricardo Montalvo Lezama

<http://turing.iimas.unam.mx/~ricardoml/>
ricardoml@turing.iimas.unam.mx

Bere Montalvo Lezama

<http://turing.iimas.unam.mx/~bereml/>
bereml@turing.iimas.unam.mx