Una Introducción al Aprendizaje Profundo Multitarea

Berenice & Ricardo Montalvo Lezama

IIMAS UNAM

github.com/richardtml/riiaa-19-dmtl

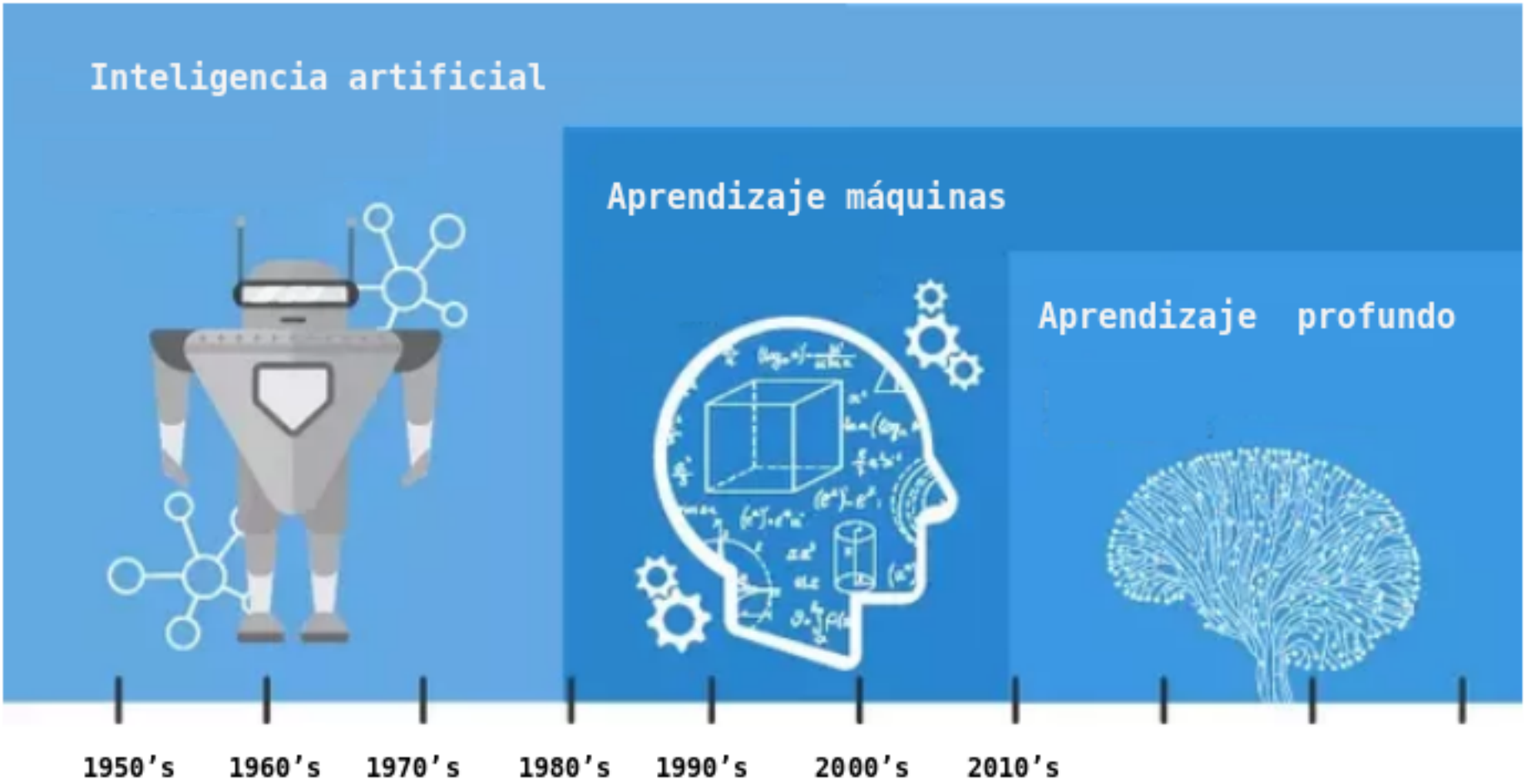
Agosto 2019

Contenidos

- Aprendizaje de profundo
  - Tipos de aprendizaje
  - Esquema de aprendizaje y modelos
  - Perceptrón multicapa
  - Redes convolucionales
- Aprendizaje multitarea
  - Aprendiendo de más fuentes
  - Aplicaciones
  - Esquema de aprendizaje y modelos
  - Problemas abiertos

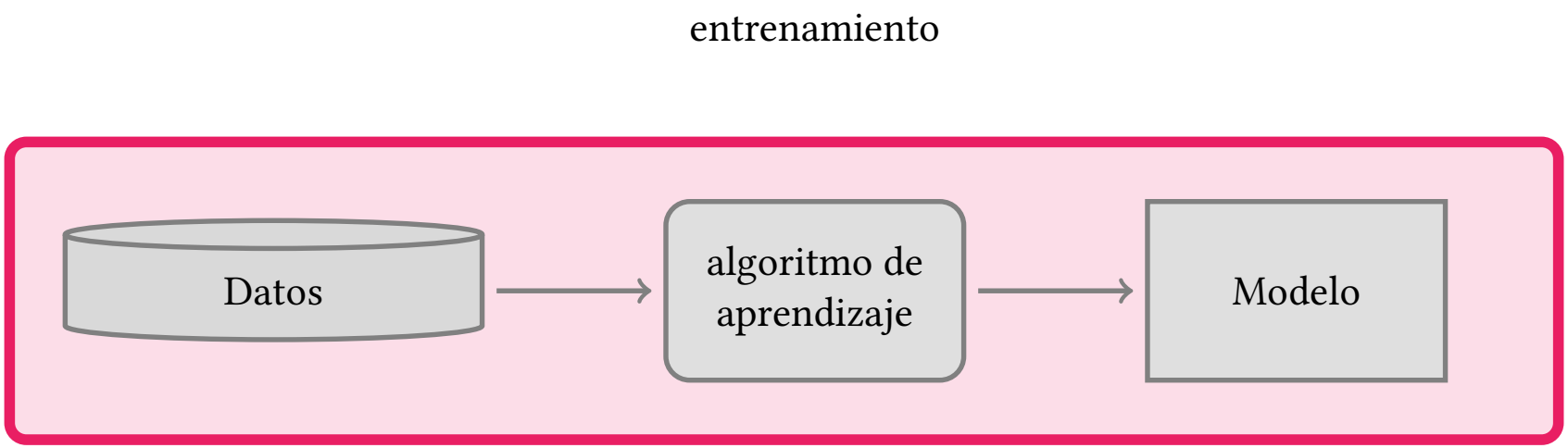
aprendizaje de profundo

IA, AM y AP


Inteligencia artificial
Aprendizaje máquinas
Aprendizaje profundo
1950's
1960's
1970's
1980's
1990's
2000's
2010's

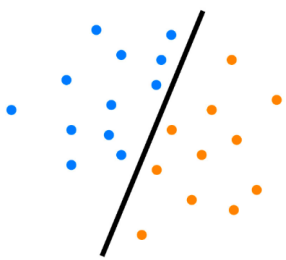
¿Cómo funciona el aprendizaje de máquinas?

- Programas que aprenden a partir de ejemplos en lugar de ser explícitamente programados.

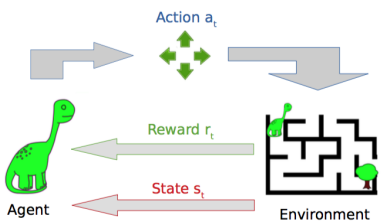


Tipos de aprendizaje de máquinas

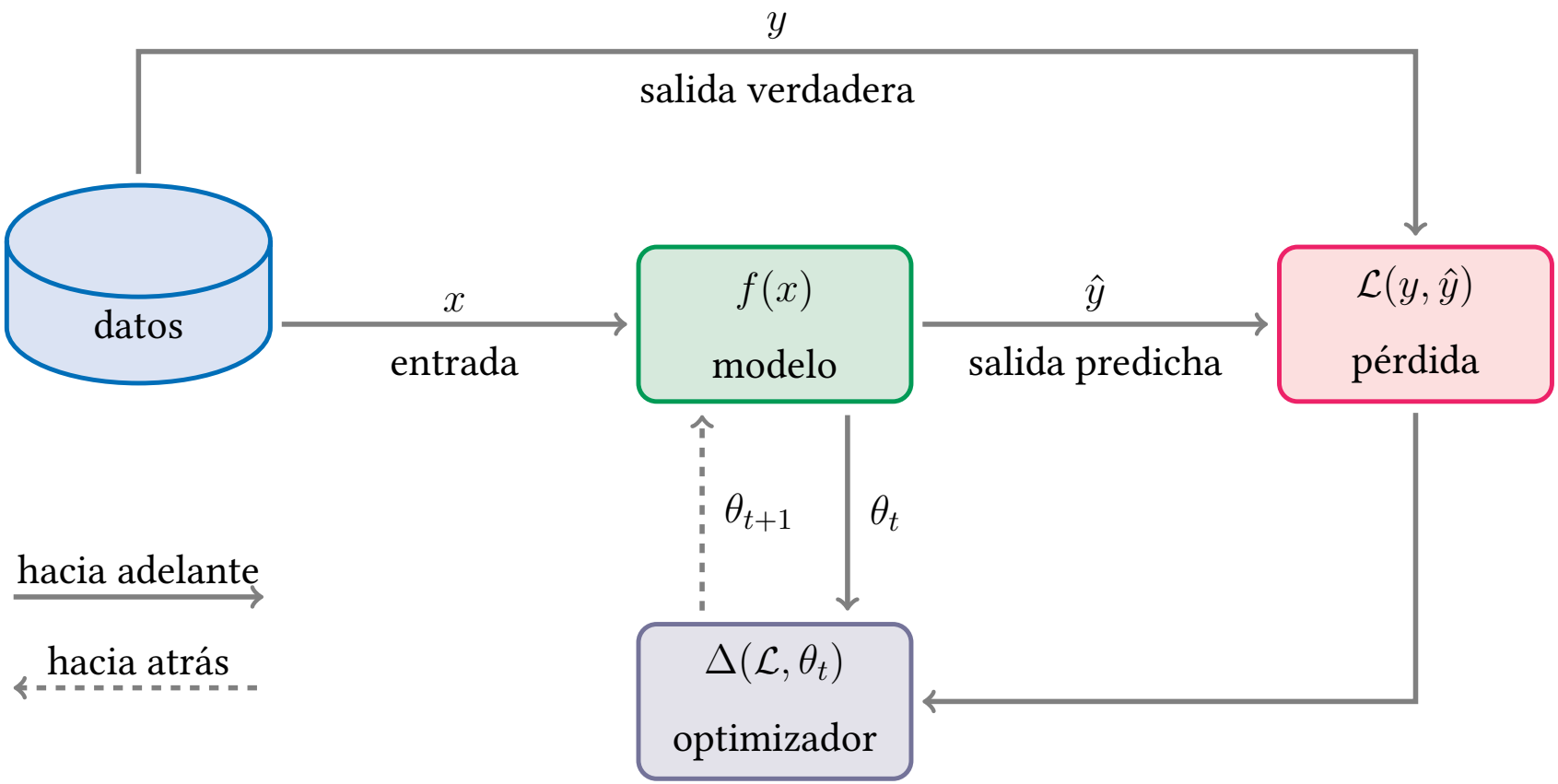
Clasificación



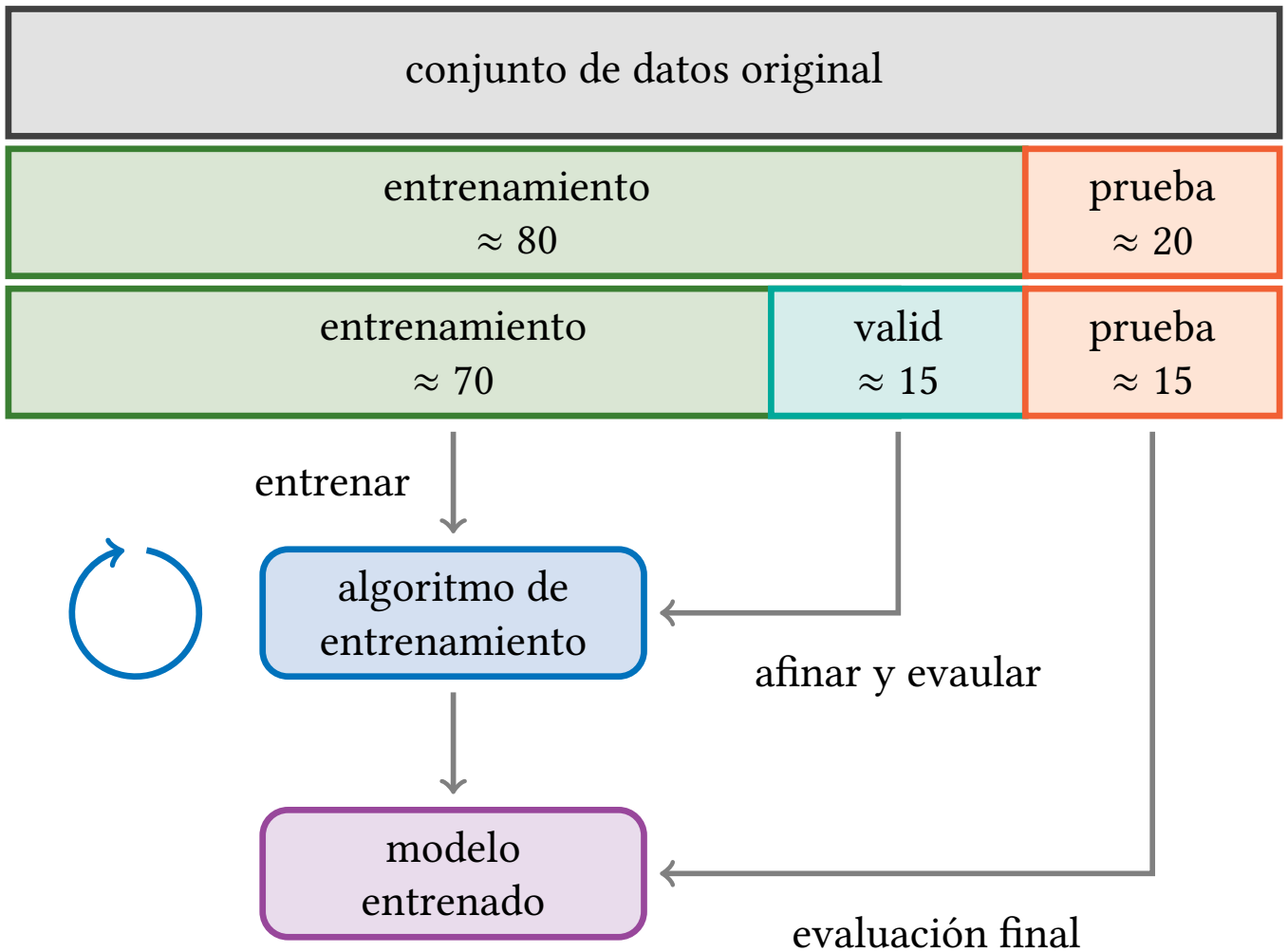
Agrupamiento



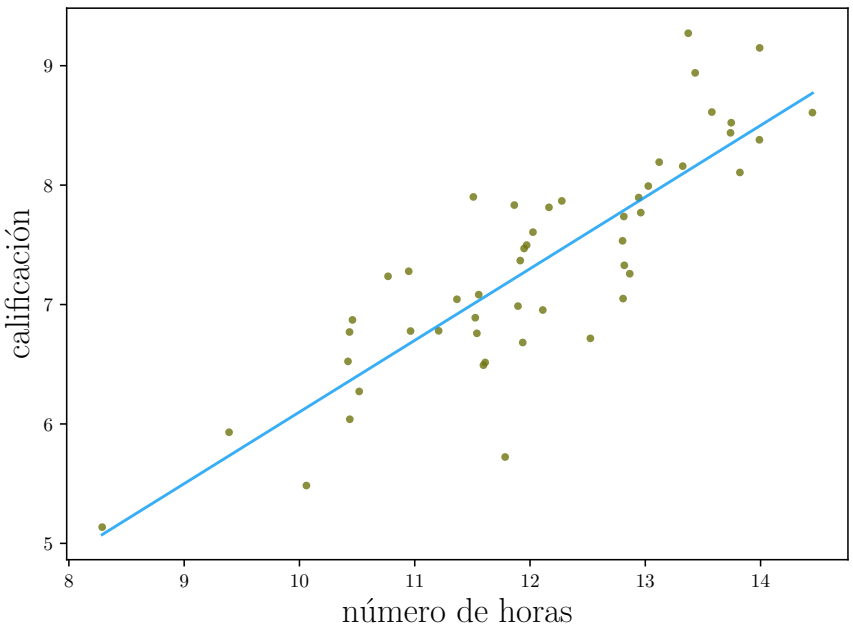
Esquema de aprendizaje: entrenamiento



Esquema de aprendizaje: conjunto de datos



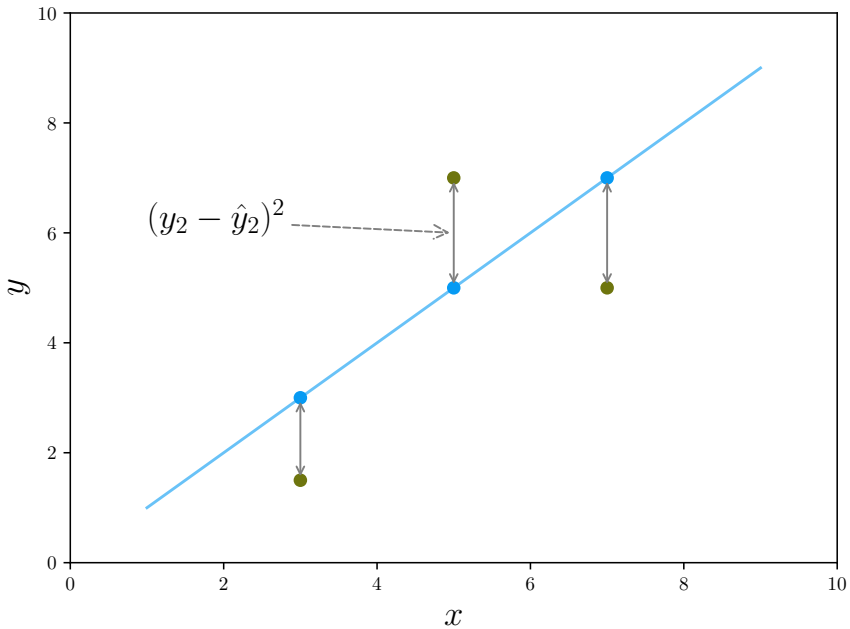
Regresión lineal simple (I)



- Modelo:

$$\hat{y} = wx + b$$

Regresión lineal simple (II)



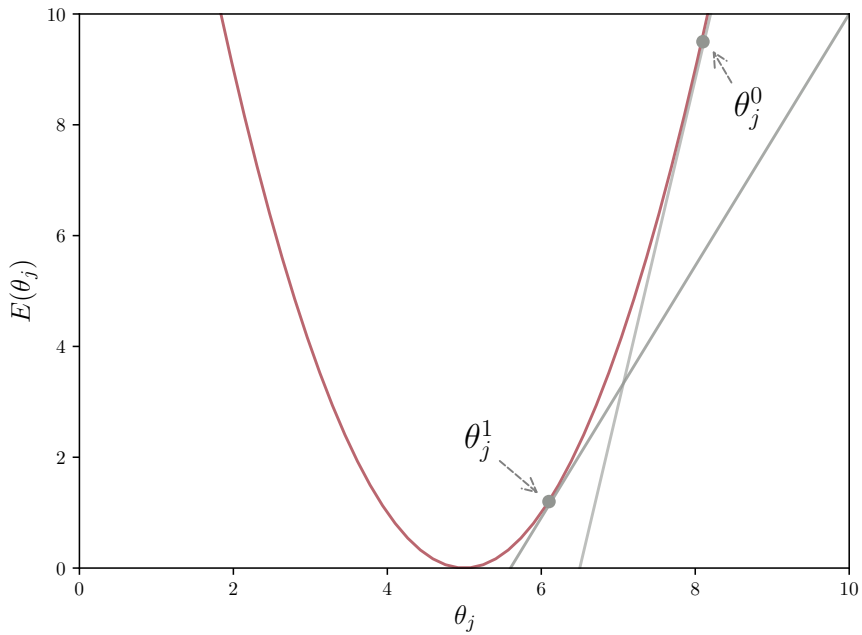
- Modelo:

$$\hat{y} = wx + b$$

- Función de error:

$$E(w, b) = \frac{1}{2} \sum_{i=1}^{n} (y - \hat{y})^2$$

Regresión lineal simple (III)



- Modelo:

$$\hat{y} = wx + b$$

- Función de error:

$$E(w, b) = \frac{1}{2} \sum_{i=1}^{n} (y - \hat{y})^2$$

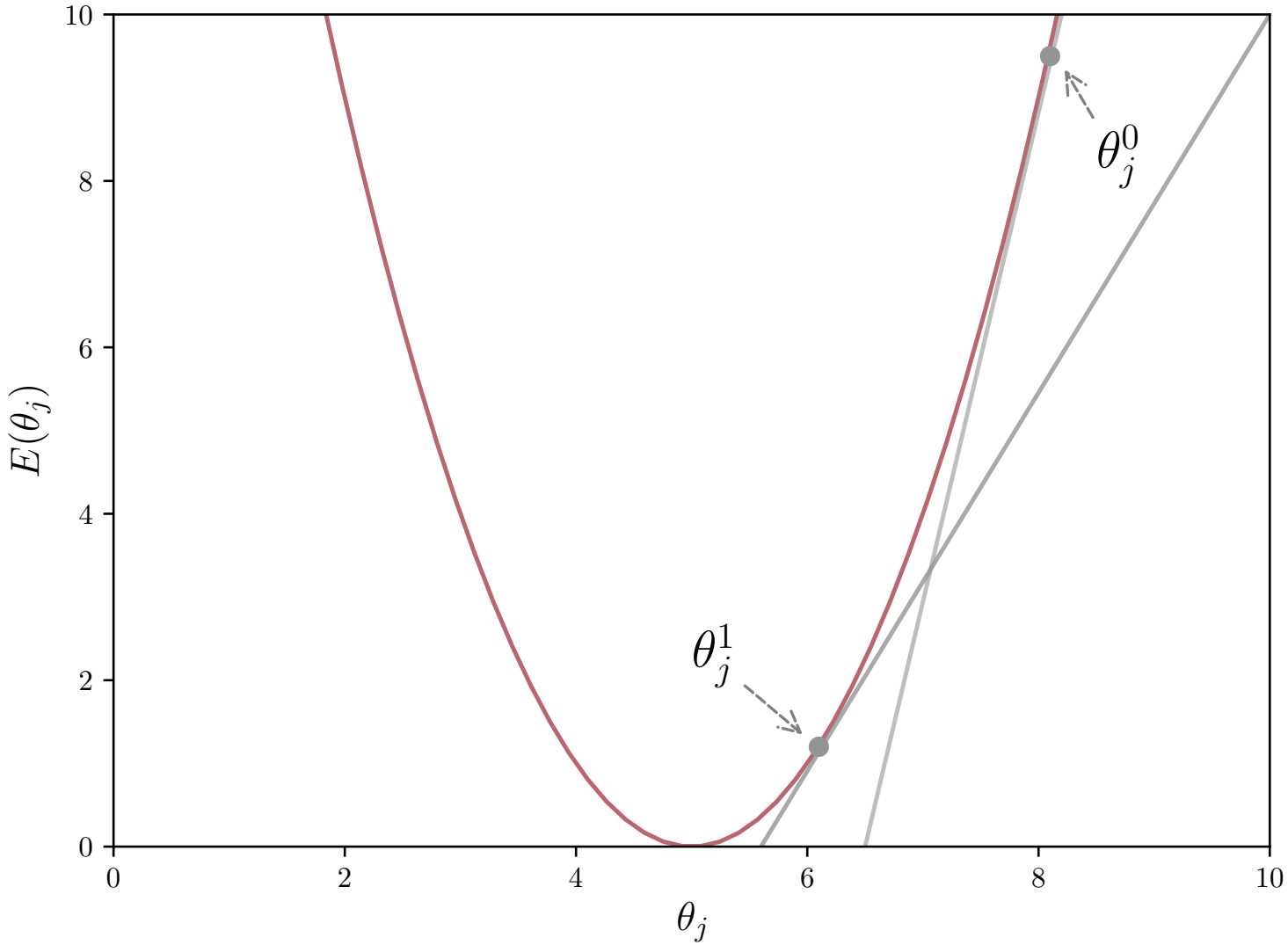
- Optimización de la función de error:

$$w := w - \alpha \frac{\partial}{\partial w} E(w, b)$$

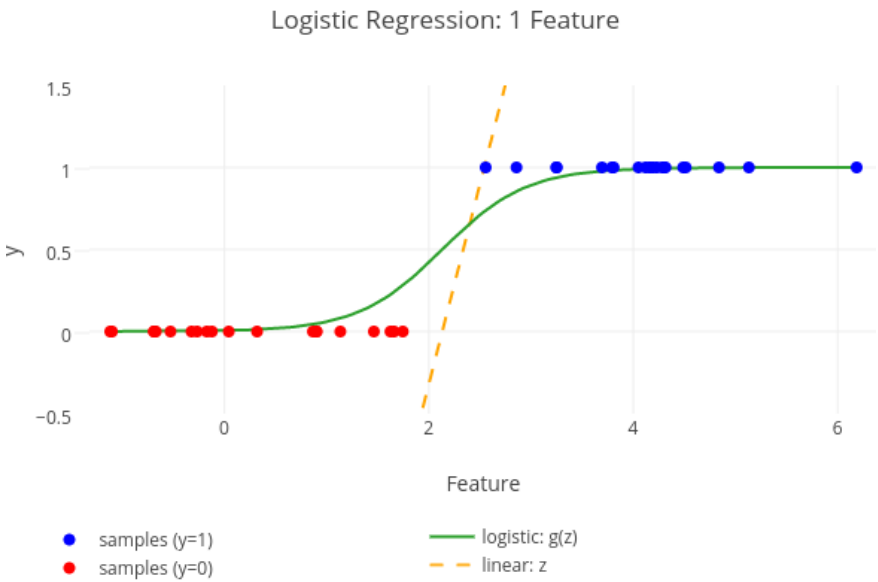
$$b := b - \alpha \frac{\partial}{\partial b} E(w, b)$$

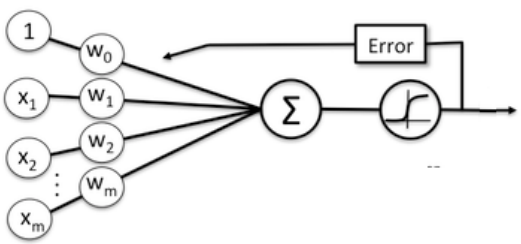
Gradiente descendente

- Algoritmo de optimización que permite encontrar el mínimo de una función de forma iterativa.



Regresión logística





- Modelo:

$$P(y|x_i, \theta) = \hat{y} = f\,\frac{1}{1+e^{wx+b}}$$

- Función de pérdida: entropía cruzada categórica:

$$E = -\frac{1}{m}\sum_{i=1}^{m}(y\log(\hat{y}) + (1-y)log(1-\hat{y}))$$

- Optimización de la función de error:

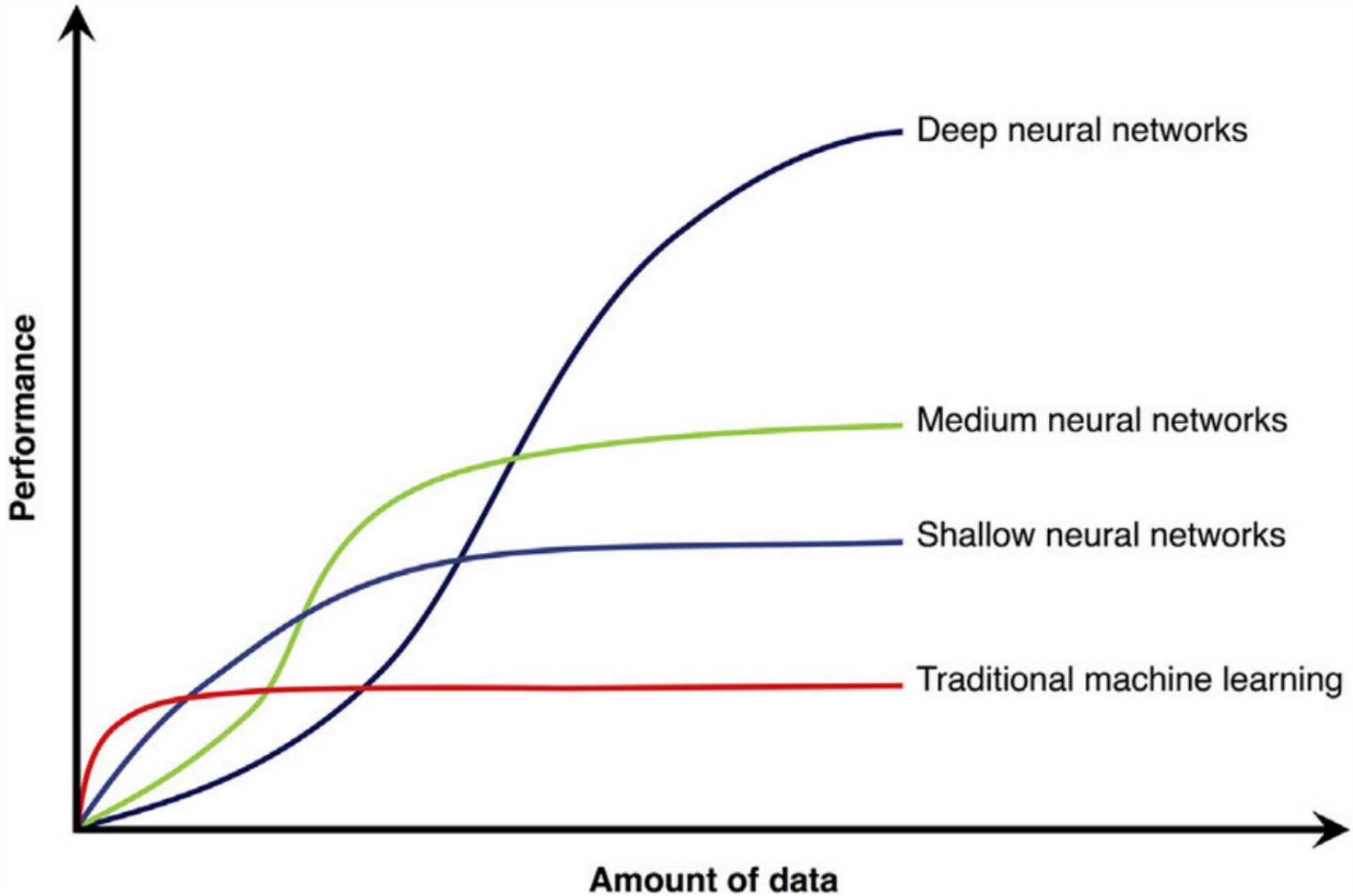
$$w := w - \alpha\,\frac{\partial}{\partial w}E(w, b)$$

$$b := b - \alpha\,\frac{\partial}{\partial b}E(w, b)$$

- Métrica: exactitud:

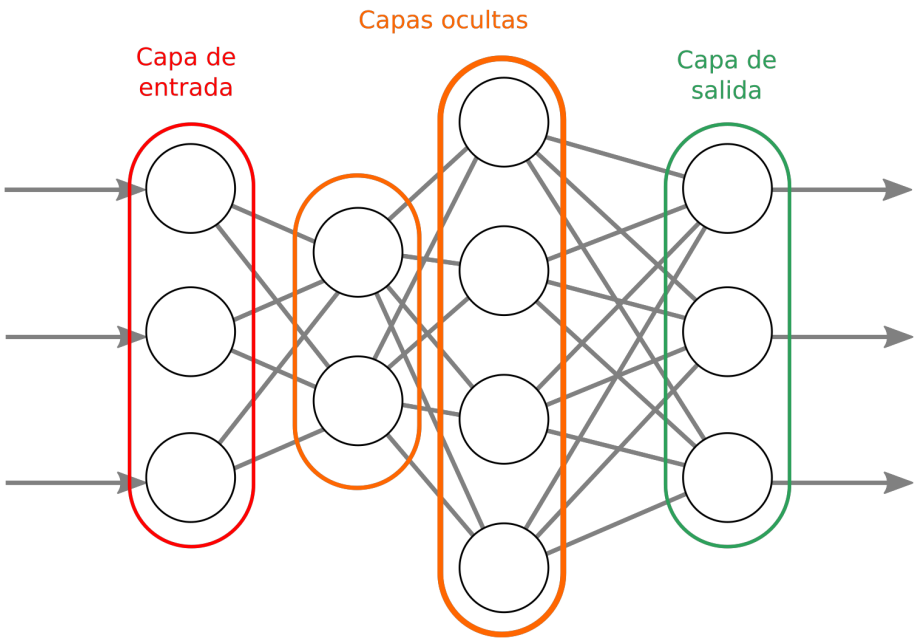
$$acc = \frac{\#\text{ correctas}}{\#\text{ todas}}$$

Limitaciones



Tang et al., *Canadian Association of Radiologists Journal 69(2)*, 2018.

Perceptrón multicapa



Entrenamiento:

- Paso 1: carga el lote de entrenamiento datos.
- Paso 2: propagación hacia adelante para obtener las predicciones y calcular el error.
- Paso 3: retropropagar el error.
- Paso 4: actualizar los pesos.

- Activación de la j-ésima neurona de la i-ésima capa:

$$a_j^i = f(\sum_{i=1}^{n} x_i w_i + b).$$

- Función de error: entropía cruzada categórica.

$$E = -\frac{1}{m} \sum_{i=1}^{m} (y \log(\hat{y}))$$

- Optimización de la función de error:

$$w := w - \alpha \frac{\partial}{\partial w} E(w, b)$$

$$b := b - \alpha \frac{\partial}{\partial b} E(w, b)$$

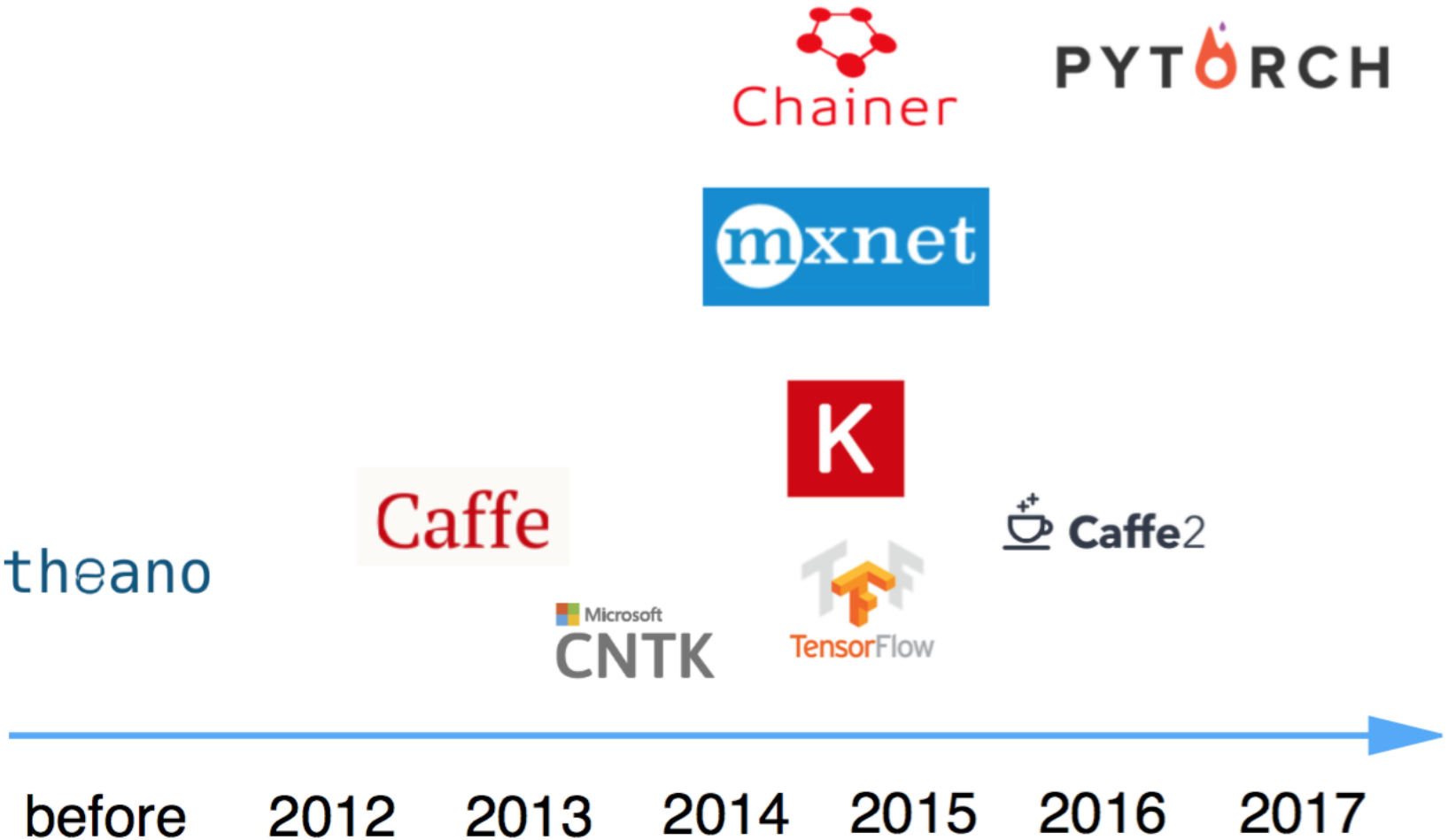
- Métrica: exactitud.

$$Ex = \frac{\#\text{ predicciones correctas}}{\#\text{ total de predicciones}}$$

Bibliotecas de AP (I)

- Modelo de programación sencillo de álgebra lineal.
- Abstracciones comunes de redes neuronales.
- Diferenciación automática.
- Optimizadores.
- Ejecución en CPU y GPU.
- Visualización, serialización, distribución, etc.

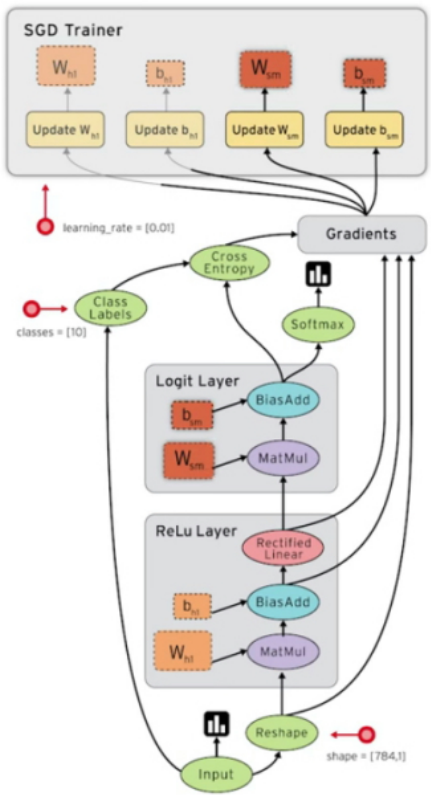
Bibliotecas de AP (II)



Gráficas de cómputo

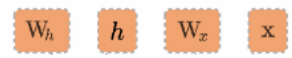
- Son gráficas dirigidas donde los nodos corresponden a operaciones y las aristas a variables.

Define-and- Run
Estática



Define-by-Run
Dinámica

A graph is created on the fly

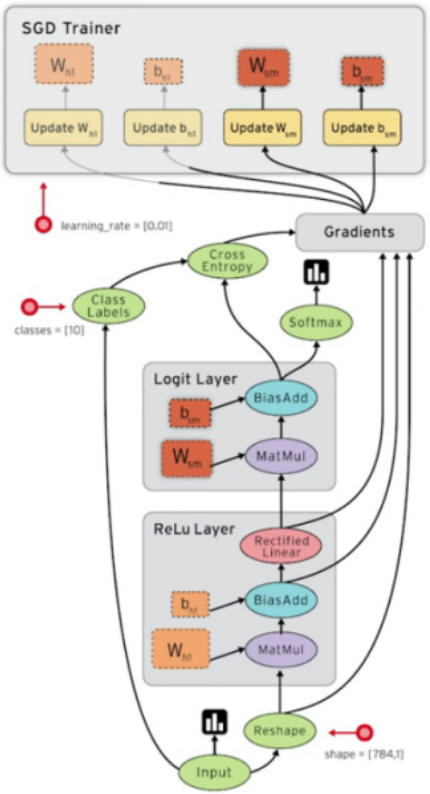



```
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))
```

Gráficas de cómputo

- Son gráficas dirigidas donde los nodos corresponden a operaciones y las aristas a variables.

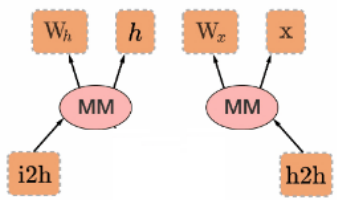
Define-and- Run
Estática



Define-by-Run
Dinámica

A graph is created on the fly

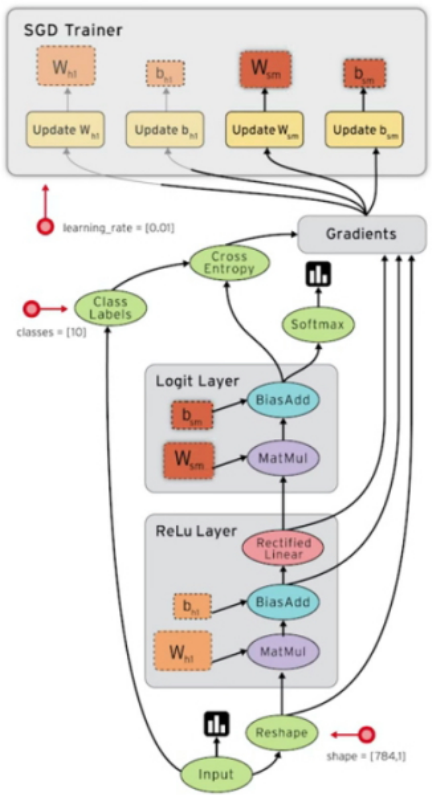
```
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))

i2h = torch.mm(W_x, x.t())
h2h = torch.mm(W_h, prev_h.t())
```



Gráficas de cómputo

- Son gráficas dirigidas donde los nodos corresponden a operaciones y las aristas a variables.

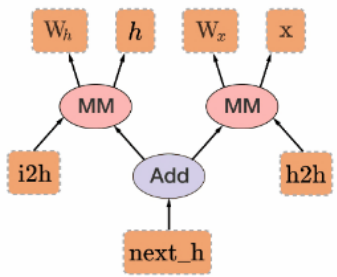
Define-and- Run
Estática



Define-by-Run
Dinámica

A graph is created on the fly

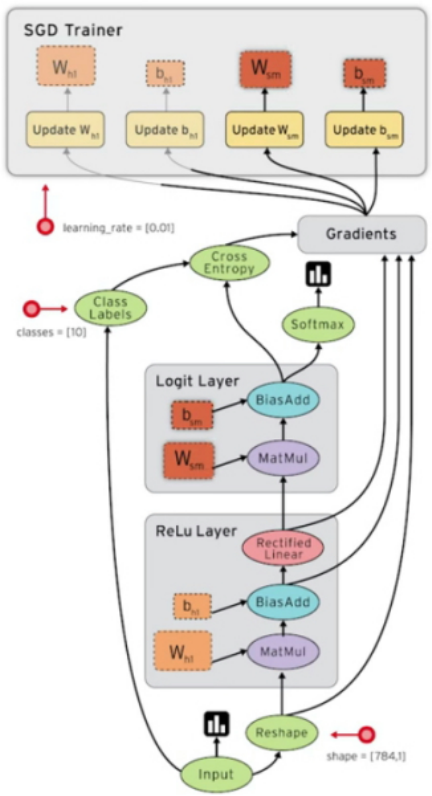
```
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))

i2h = torch.mm(W_x, x.t())
h2h = torch.mm(W_h, prev_h.t())
next_h = i2h + h2h
```



Gráficas de computo

- Gráficas dirigidas donde los nodos son operaciones y las aristas son variables.

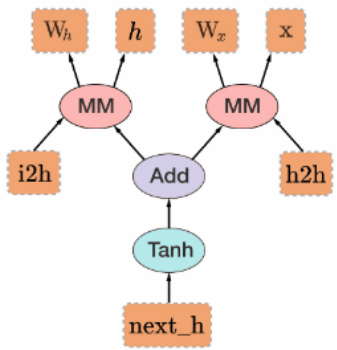
Define-and- Run
Estática



Define-by-Run
Dinámica

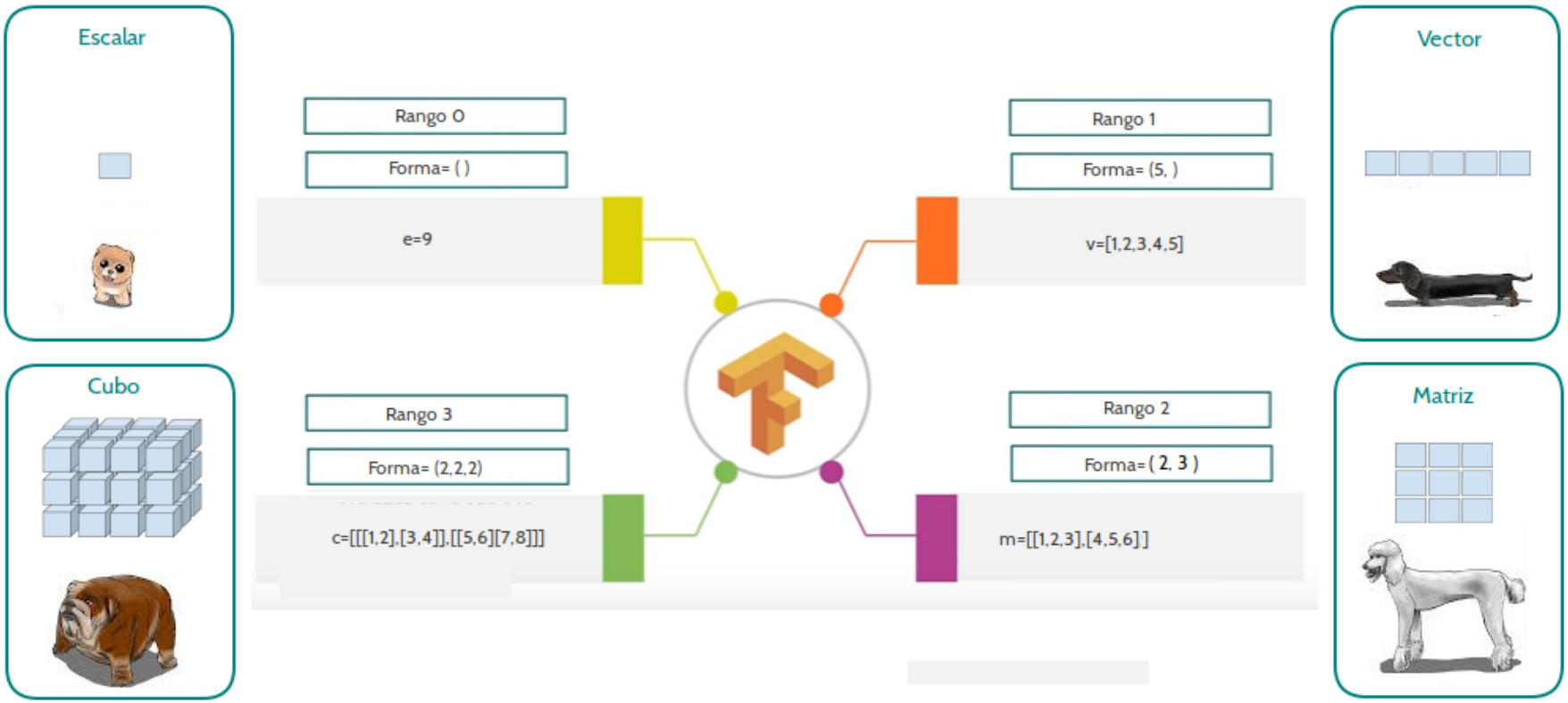
A graph is created on the fly

```
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))

i2h = torch.mm(W_x, x.t())
h2h = torch.mm(W_h, prev_h.t())
next_h = i2h + h2h
next_h = next_h.tanh()
```



Tensores

- Arreglos multidimencionales con rastreo de flujo de cómputo.



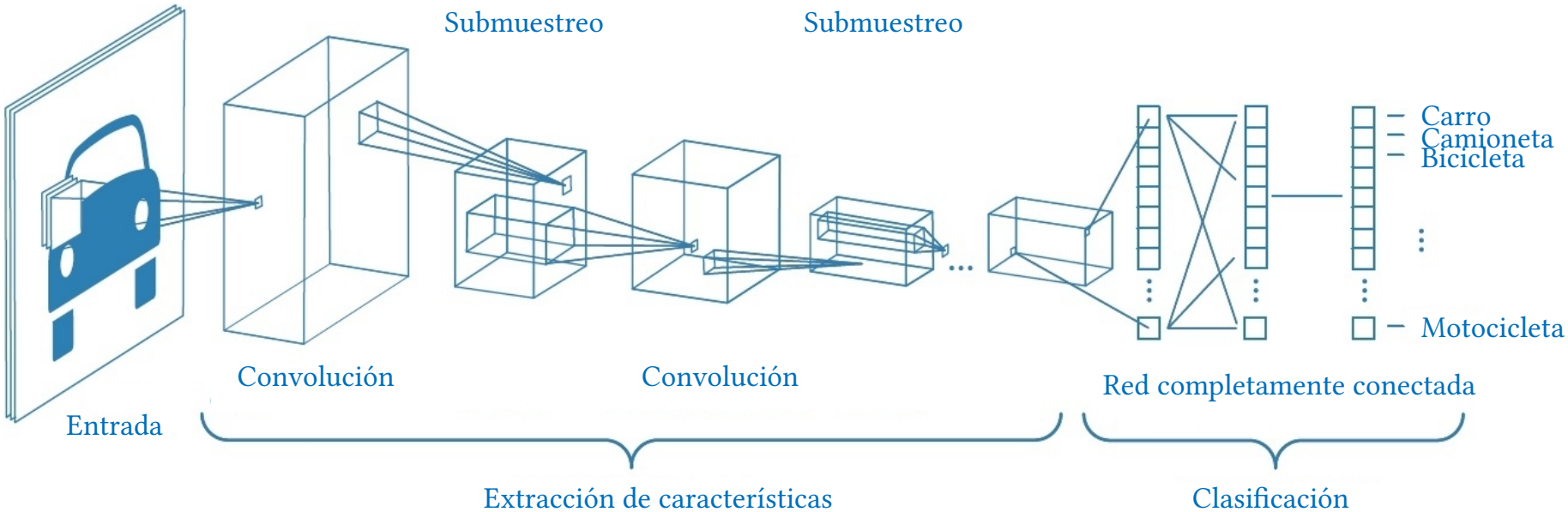
Interfaces de TensorFlow 2.0

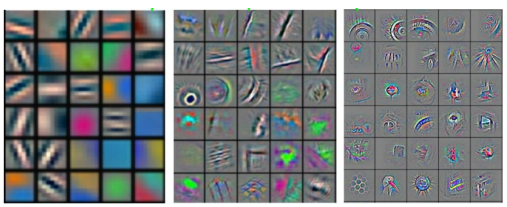
- Alto nivel.
  - Apilando capas con tf.keras.models.Sequential.
- Medio nivel.
  - Heredando de tensorflow.keras.Model.
- Bajo nivel.
  - Usando primitivas de TensorFlow.

¡tiempo de codear!

1_mlp.ipynb

Redes convolucionales





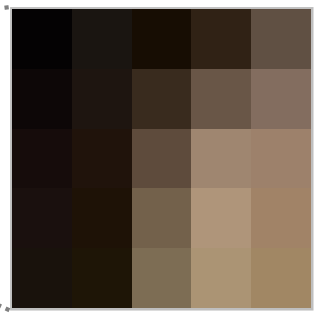
Zeiler et al. *Visualizing and Understanding Convolutional Networks*. 2013.

He et al. *Deep Residual Learning for Image Recognition*. 2015.

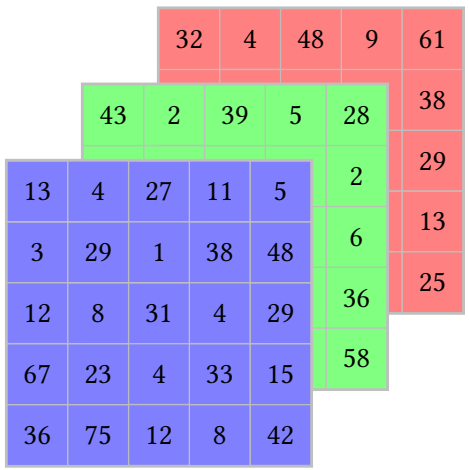
Representación de imágenes

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

Pasita $224 \times 224 \times 3$



región $5 \times 5 \times 3$

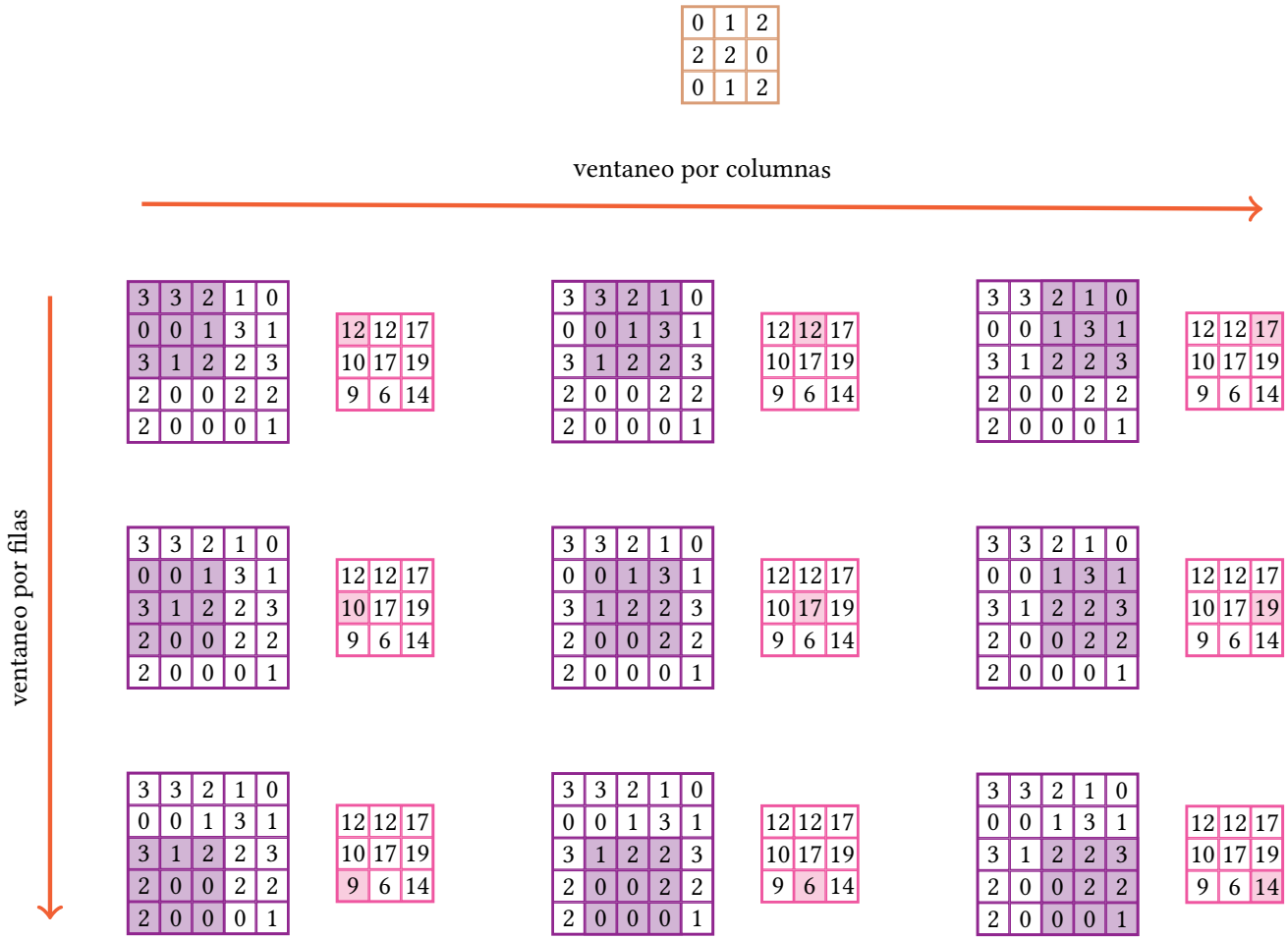


canales RGB

- Dimensiones: MINIST 28×28, CIFAR-10 32×32, ImageNet 256×256.

Convolución

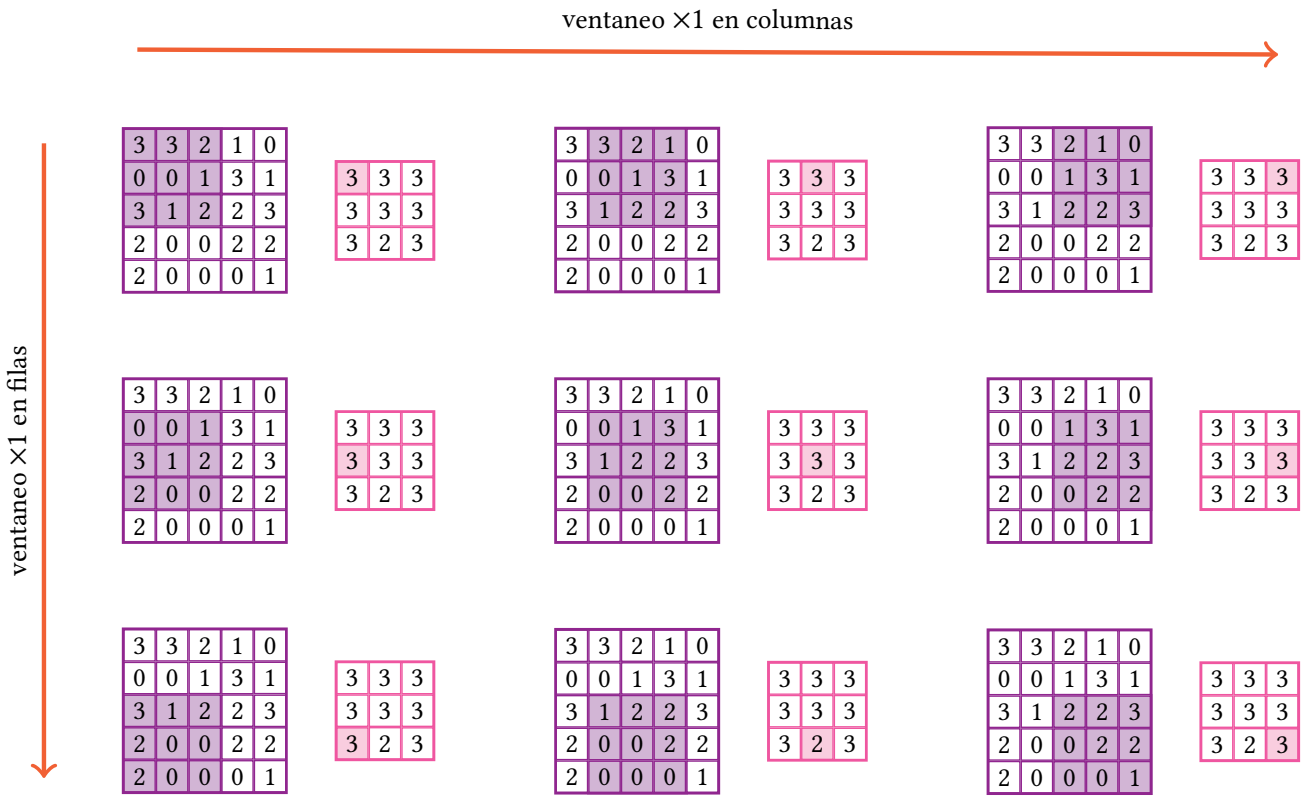
- Explota la conectividad local de pequeñas regiones para detectar características.



Convolución: entrada 5×5, salida 3×3, filtro 3x3.

Muestreo

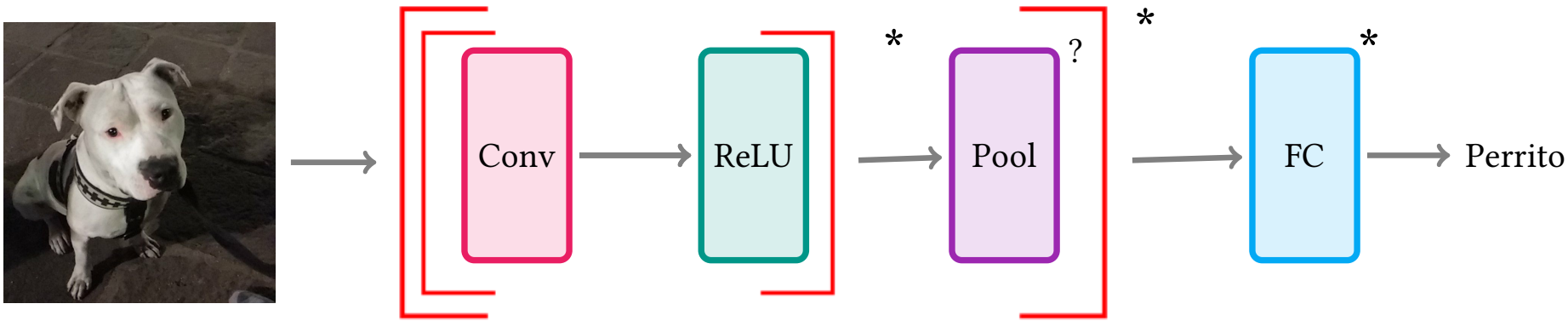
- Reducen el tamaño de la entrada mediante el uso de alguna función para resumir subregiones.



Muestreo máximo: entrada 5×5, salida 3×3, paso 1x1.

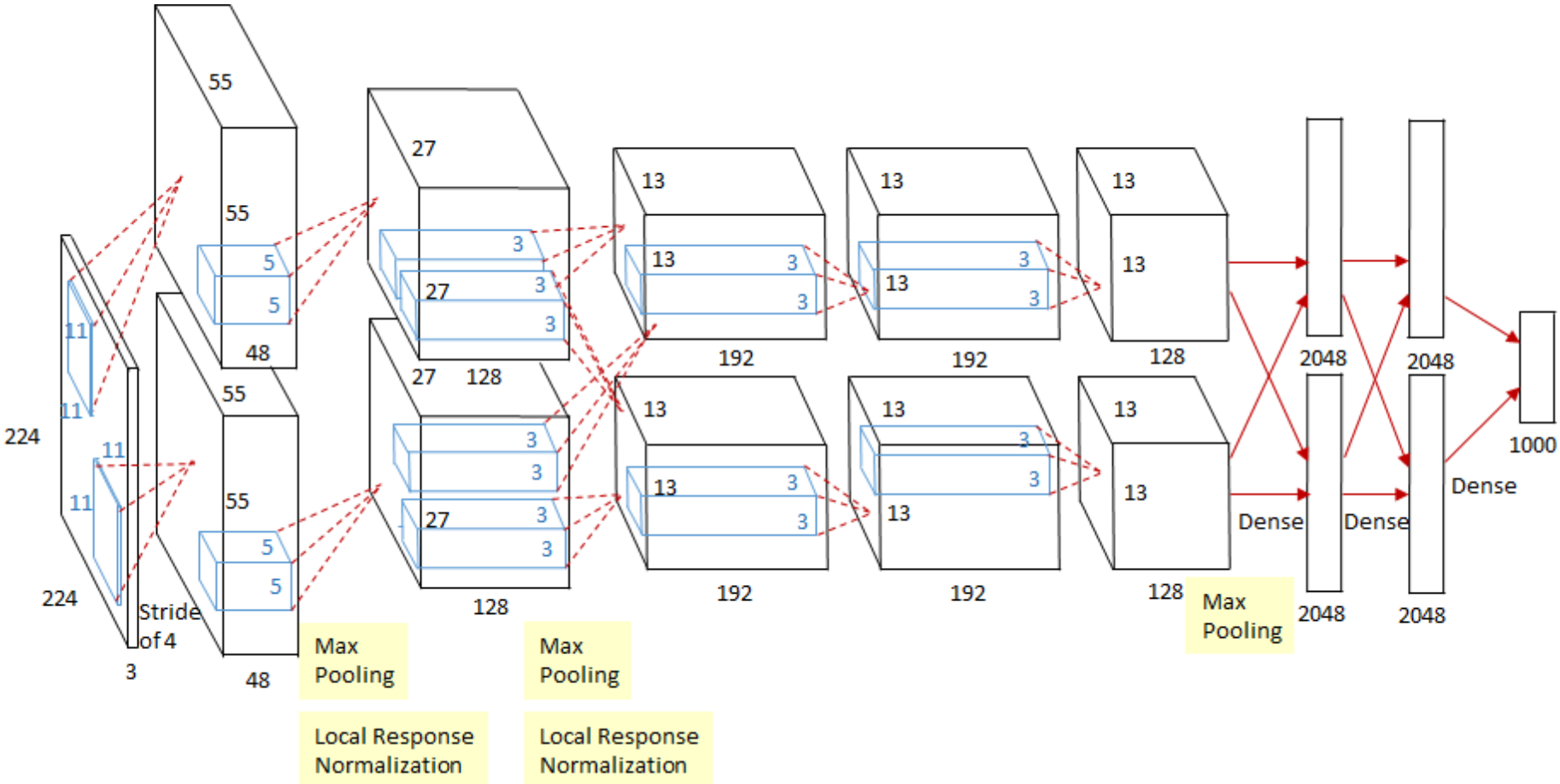
Arquitecturas de CNNs

- Aumentar canales con capas convolucionales, reducir dimenciones con capas de muestreo.



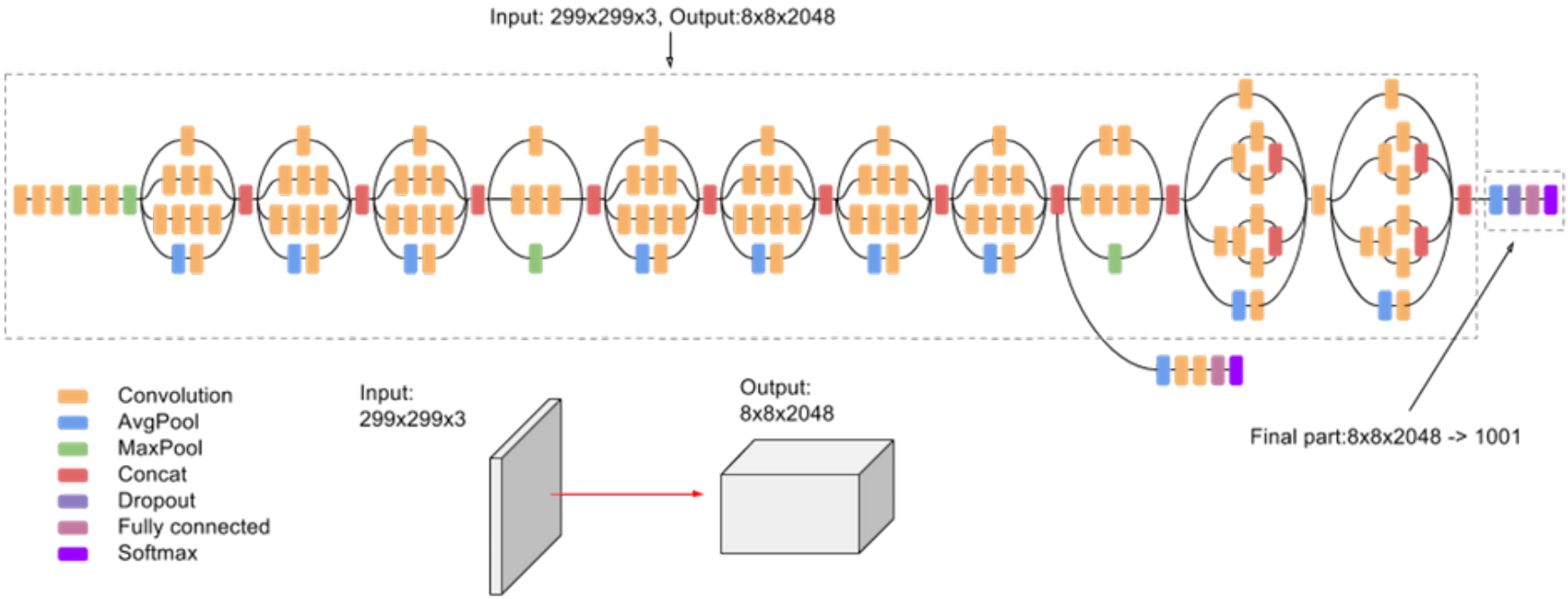
Dumoulin et al. *A guide to convolution arithmetic for deep learning*. 2018.

AlexNet



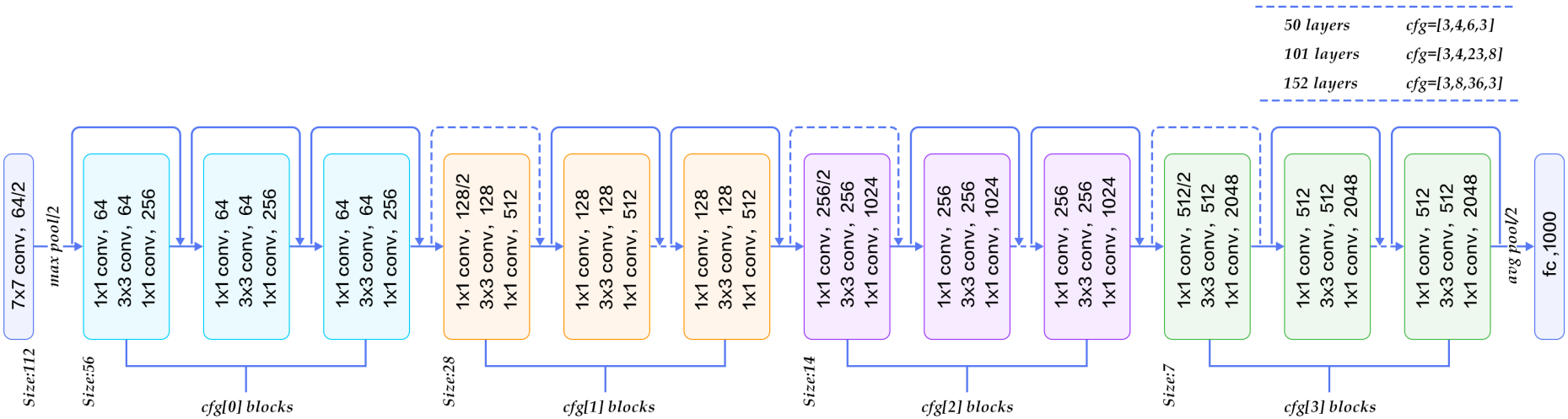
Krizhevsky et al. *ImageNet Classification with Deep Convolutional Neural Networks.* 2012.

InceptionV3



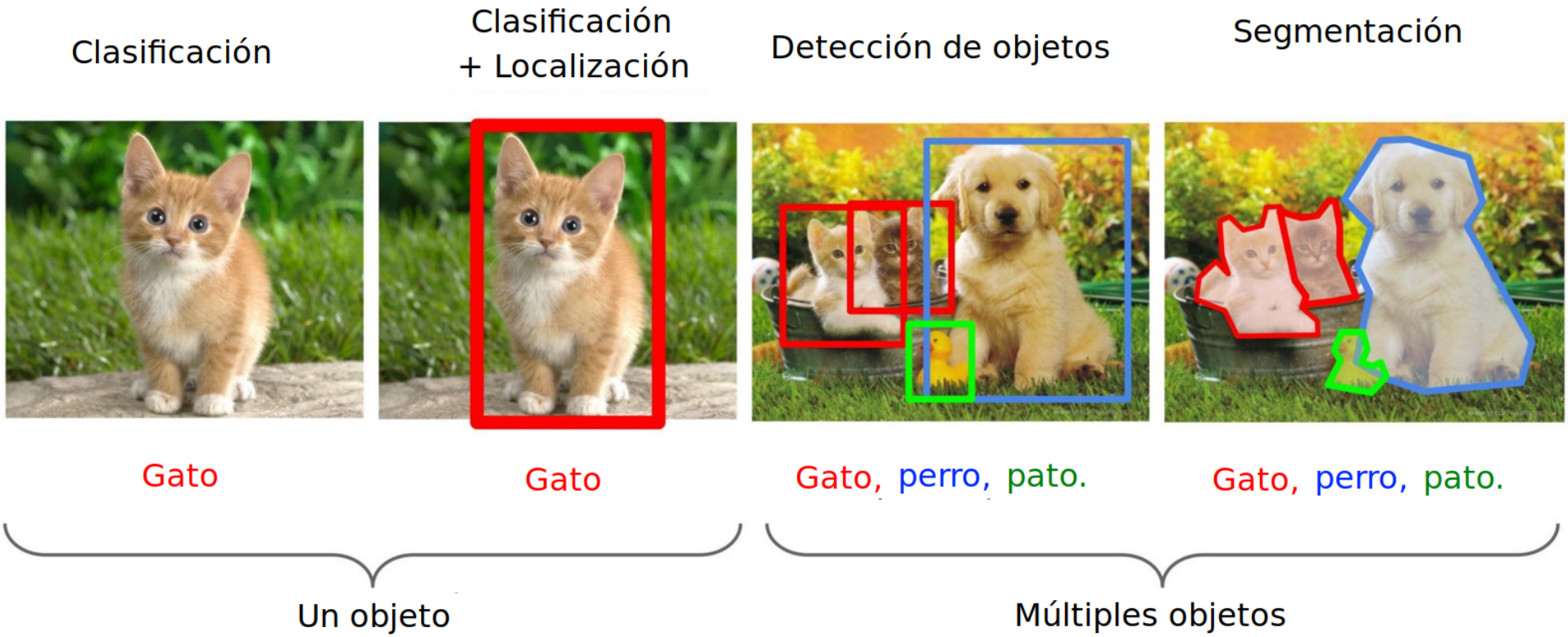
Szegedy et al. *Rethinking the Inception Architecture for Computer Vision*. 2015.

ResNet



He et al. *Deep Residual Learning for Image Recognition.* 2015.

Tareas en análisis de imágenes

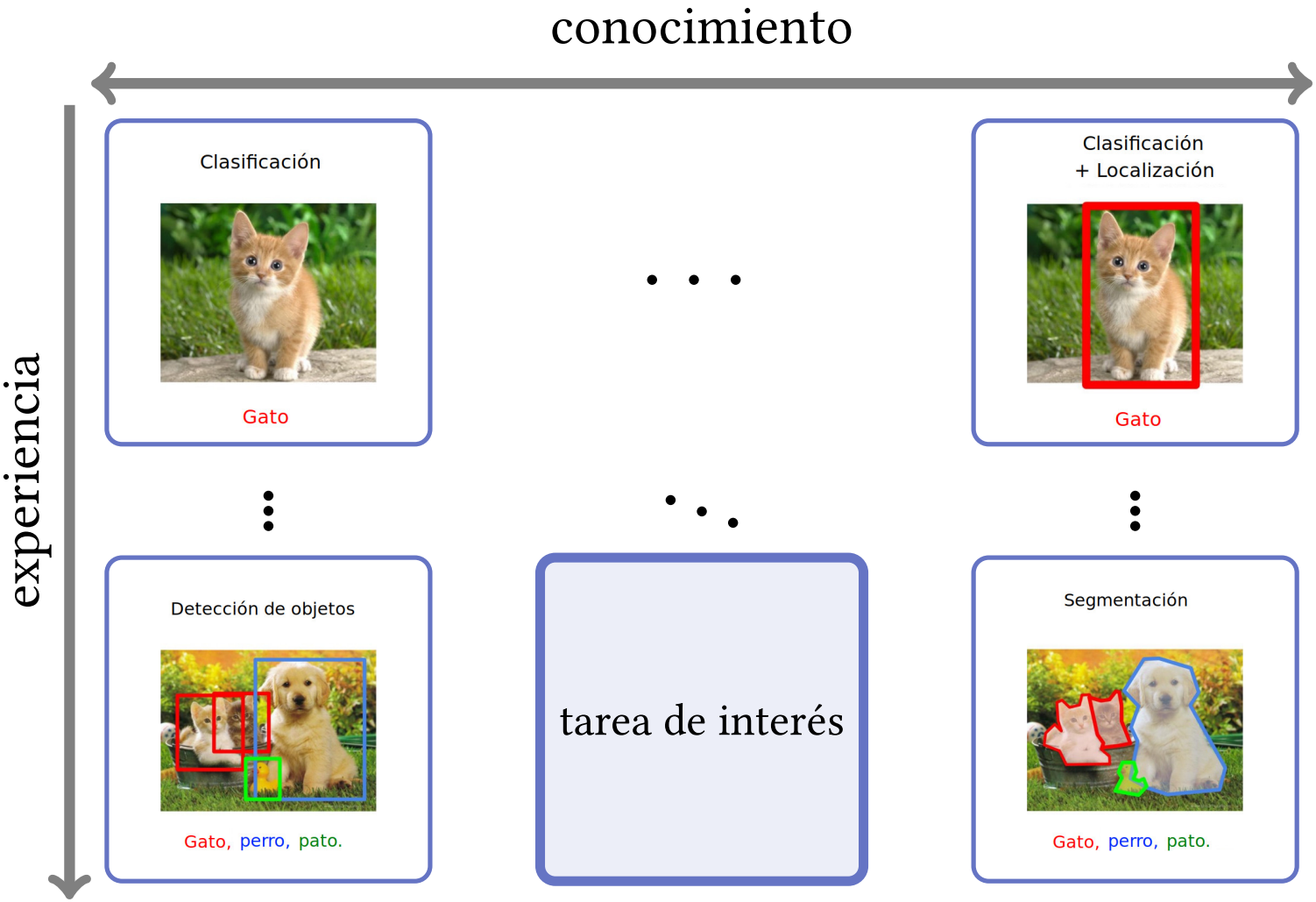


¡tiempo de codear!

2_cnn.ipynb

aprendizaje multitarea

¿Cómo podemos aprender más y mejor?



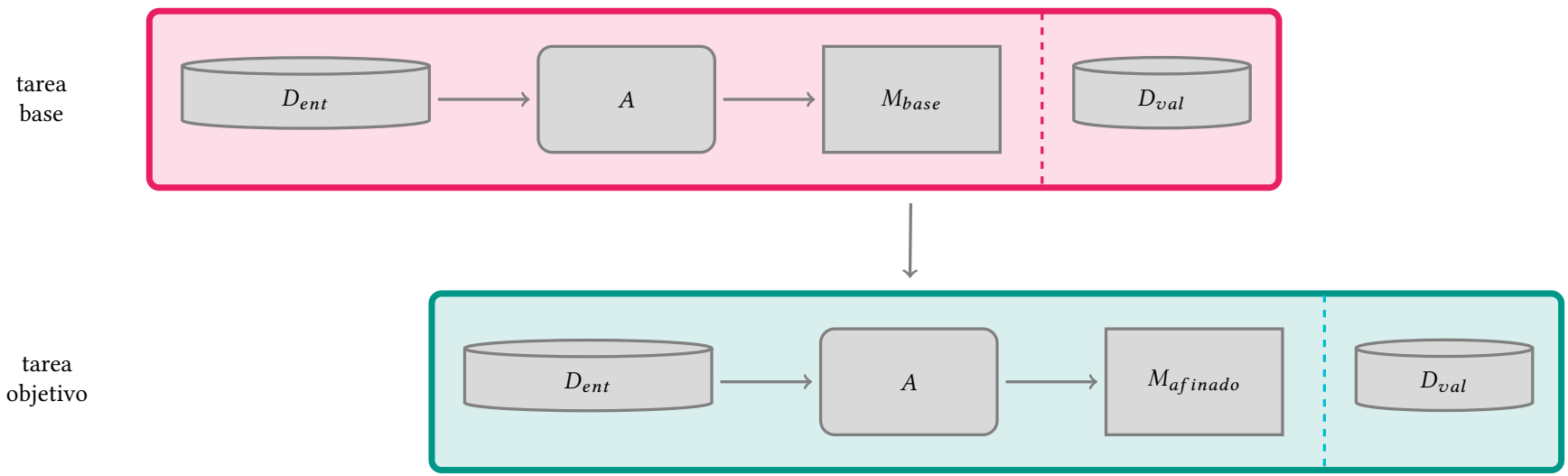
Formulaciones

- Tranferencia de conocimiento
- Aprendizaje multitarea
- Meta aprendizaje
- Aprendizaje a lo largo de la vida
- Aprendizaje multimodal
- Y otros mas!

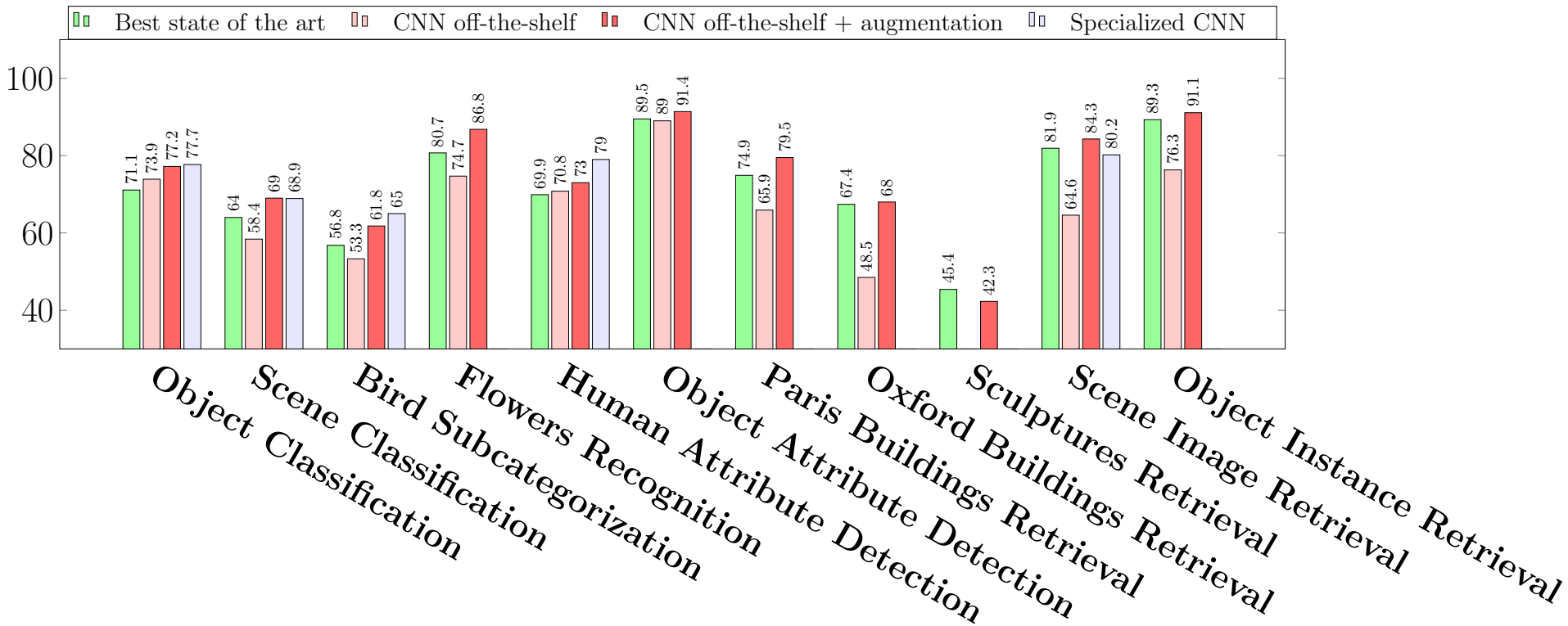
Ying et al. *Transfer Learning via Learning to Transfer*, 2018.

Transferencia de conocimiento (I)

- Aprovechar el conocimiento de una tarea base en un tarea objetivo.



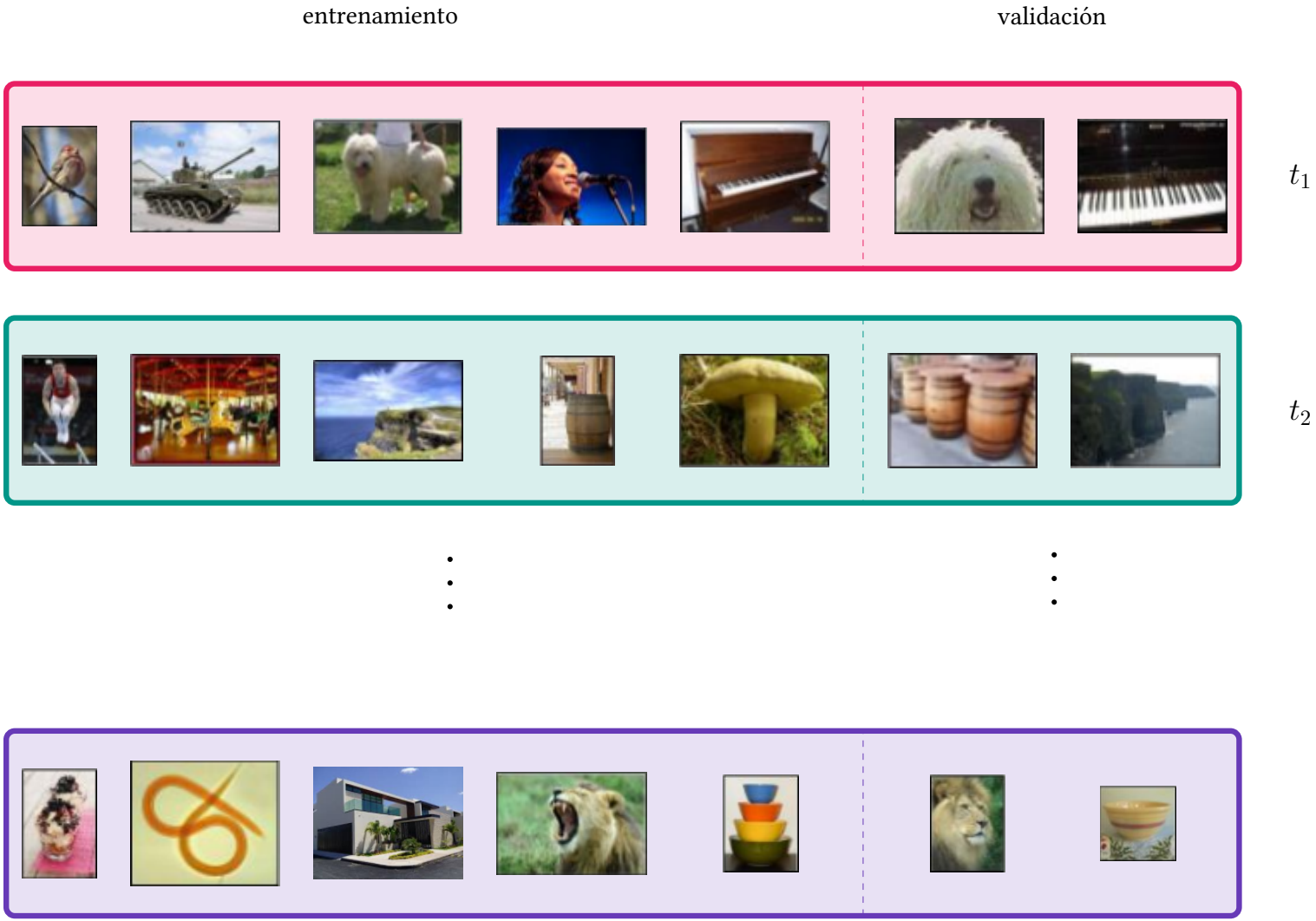
Transferencia de conocimiento (II)



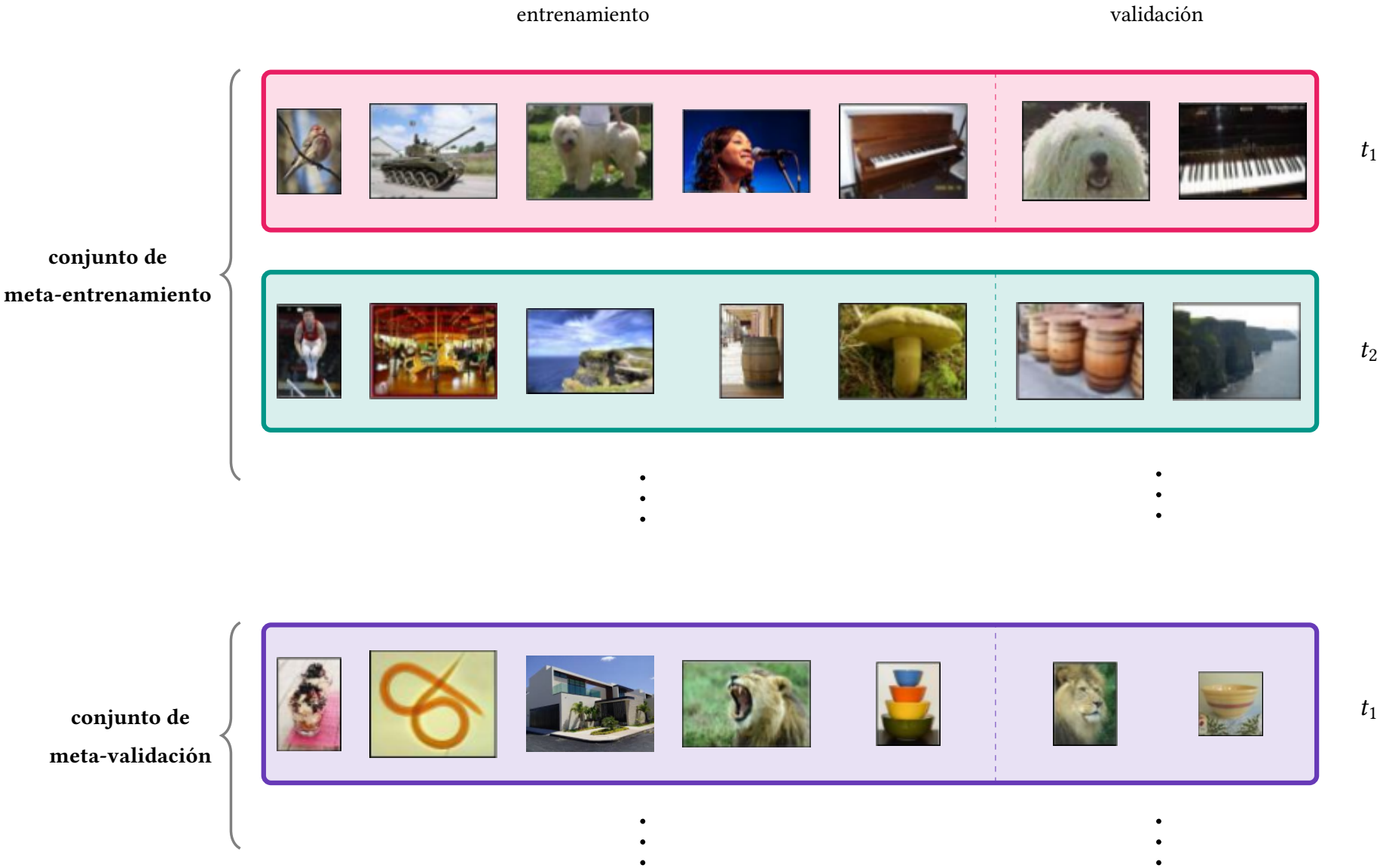
[1].

[1] Yosinski et al. *How transferable are features in deep neural networks?* 2014.

Aprendizaje multitarea (AMT)



Meta aprendizaje



AMT en procesamiento de lenguaje natural

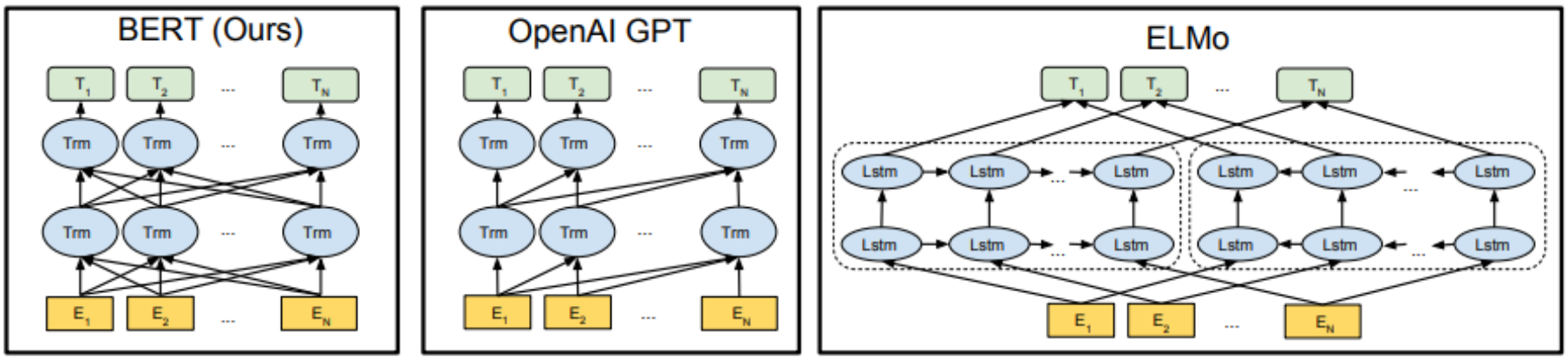
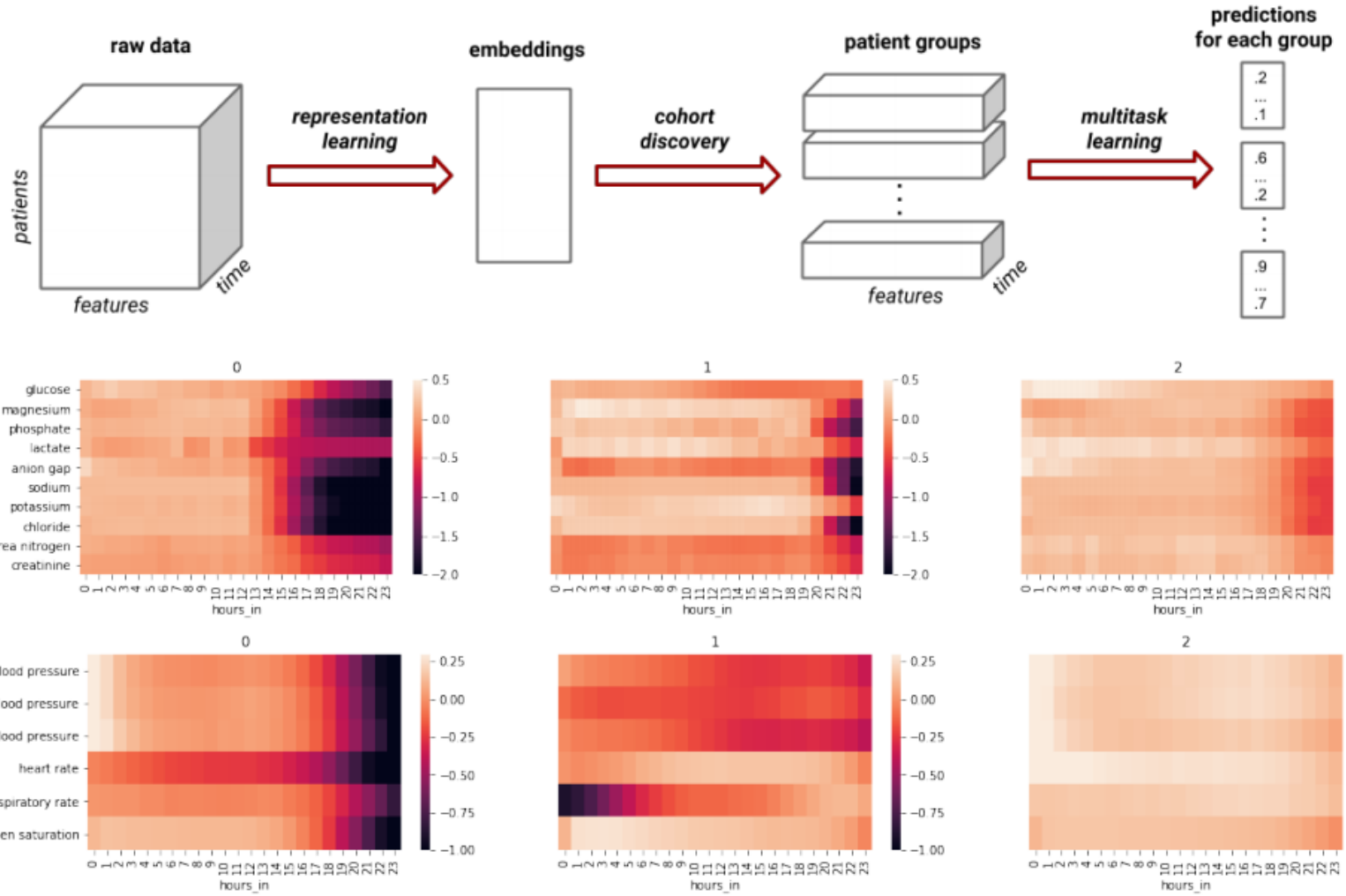


Figure 1: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM to generate features for downstream tasks. Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

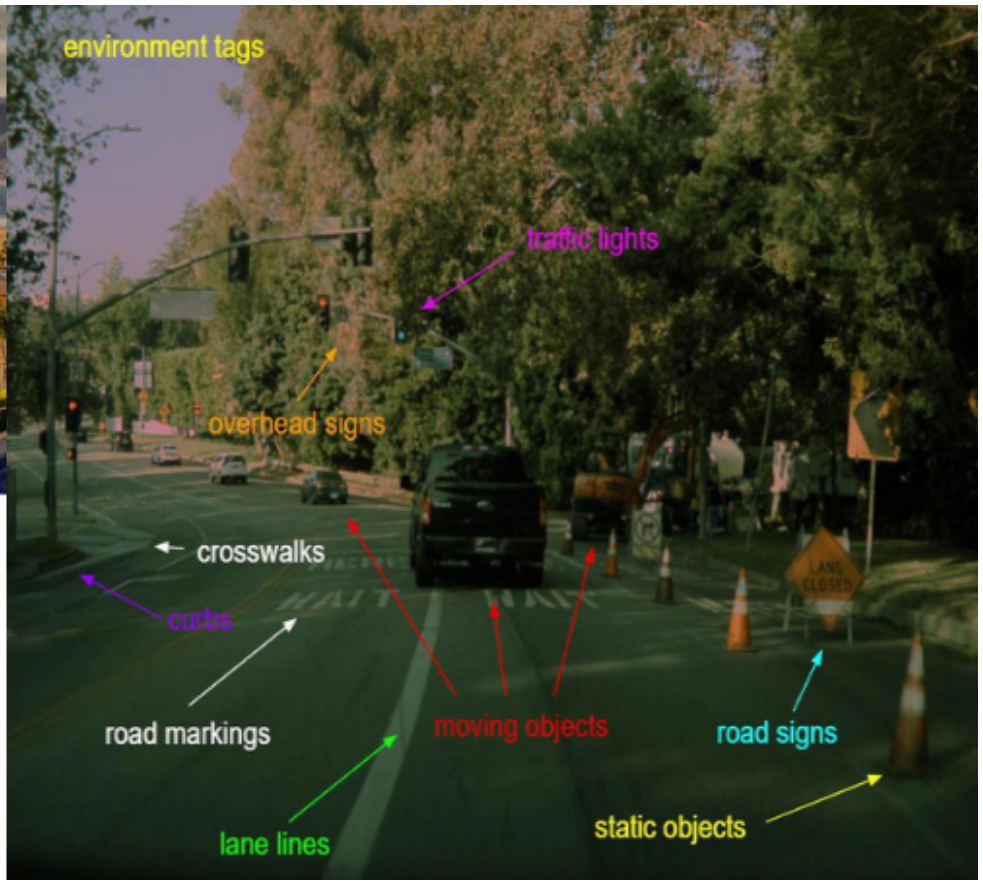
J. Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* 2018.

AMT en médicina



Suresh, Gong, Guttag. *Learning Tasks for Multitask Learning: Heterogenous Patient Populations in the ICU*. 2018.

AMT en visión



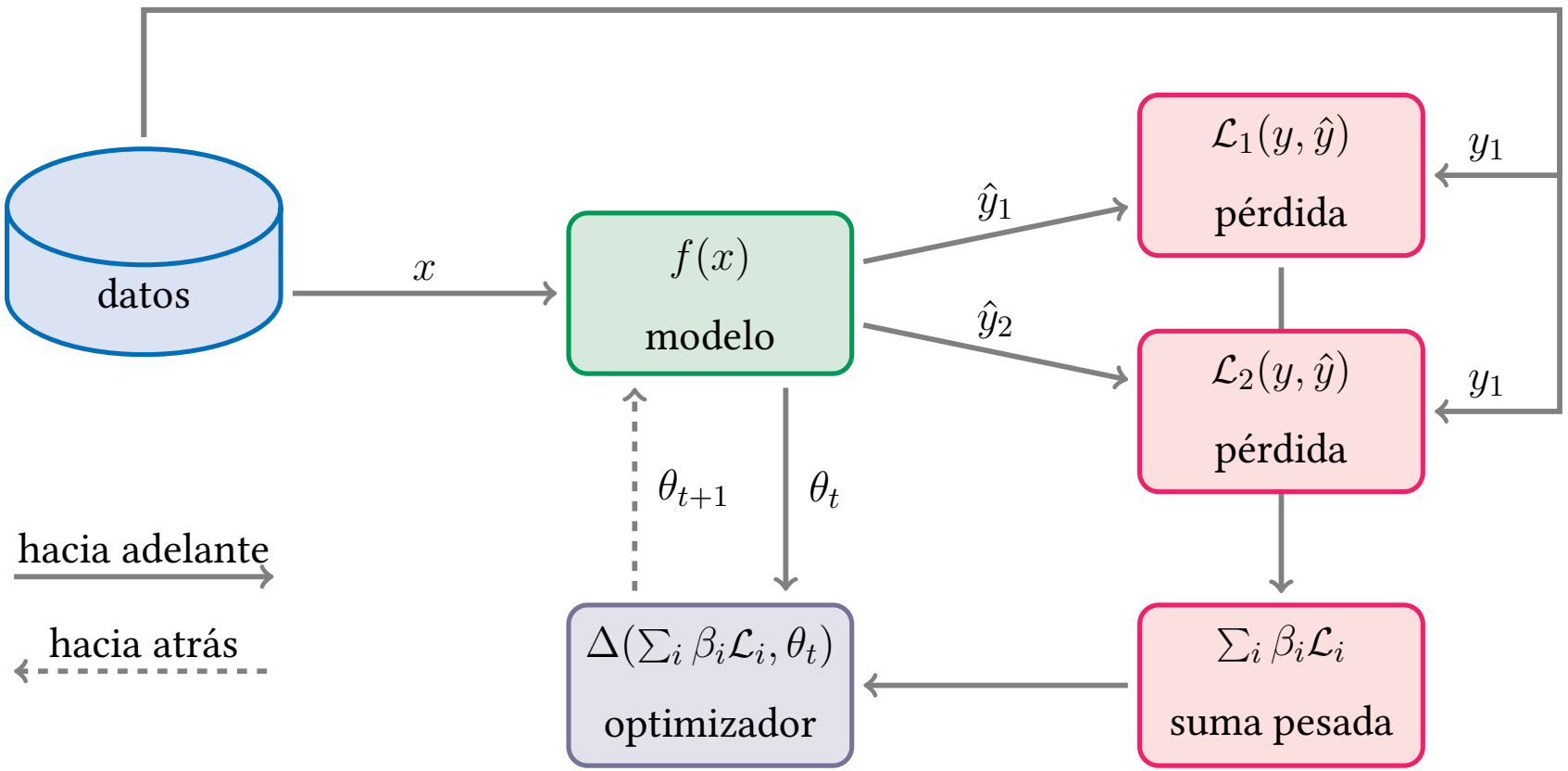
https://slideslive.com/38917690/multitask-learning-in-the-wilderness

¿Por qué funciona aprendizaje multitarea?

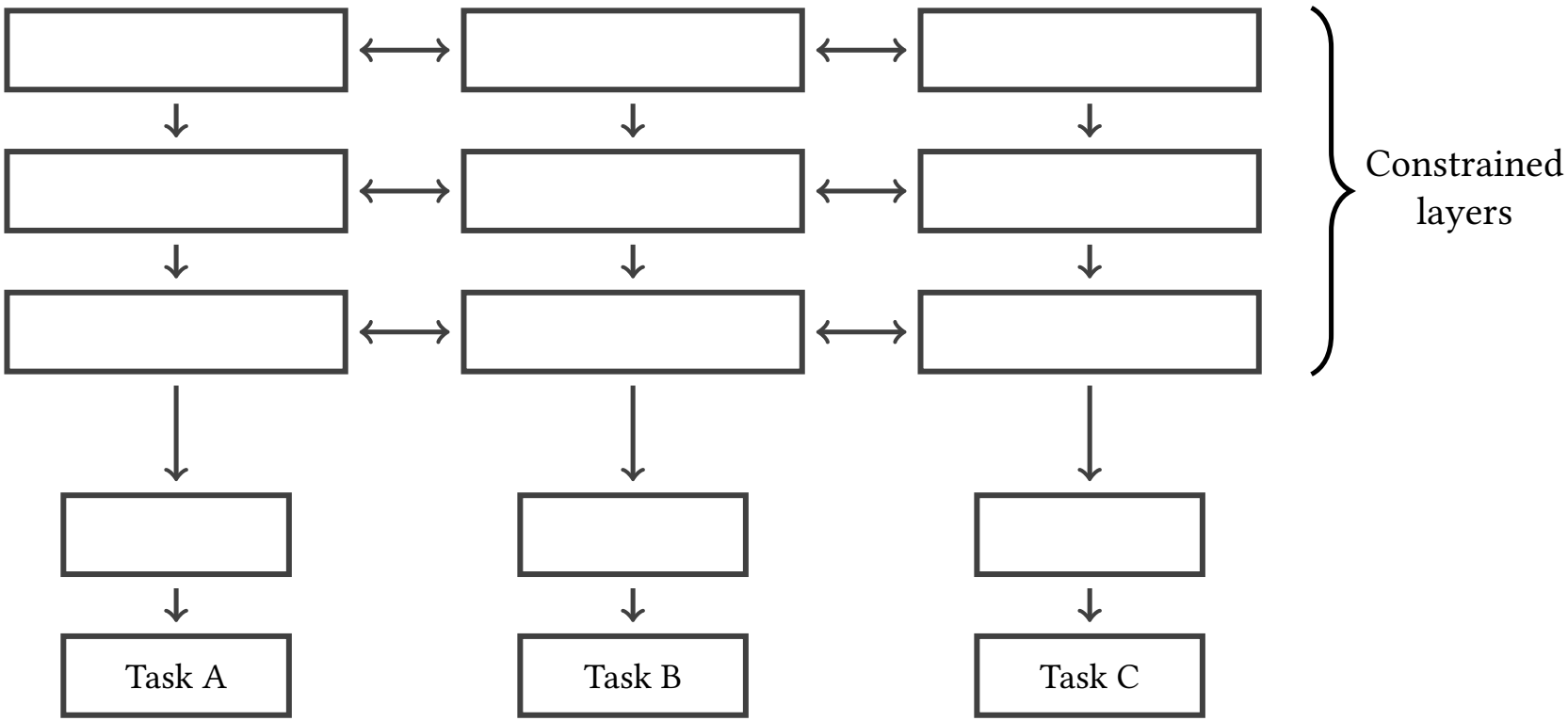
- Aumentado de datos implícito.
  - Más tareas más datos.
- Atención focalizada.
  - Consentrarse en lo importante.
- Chismorreo.
  - Aprender de otras lo que es dificil para una.
- Sesgo de representación.
  - Preferir representaciones que beneficien a todas.

Sebastian Ruder. *An Overview of Multi-Task Learning in Deep Neural Networks*, 2017.

Esquema de aprendizaje multitarea

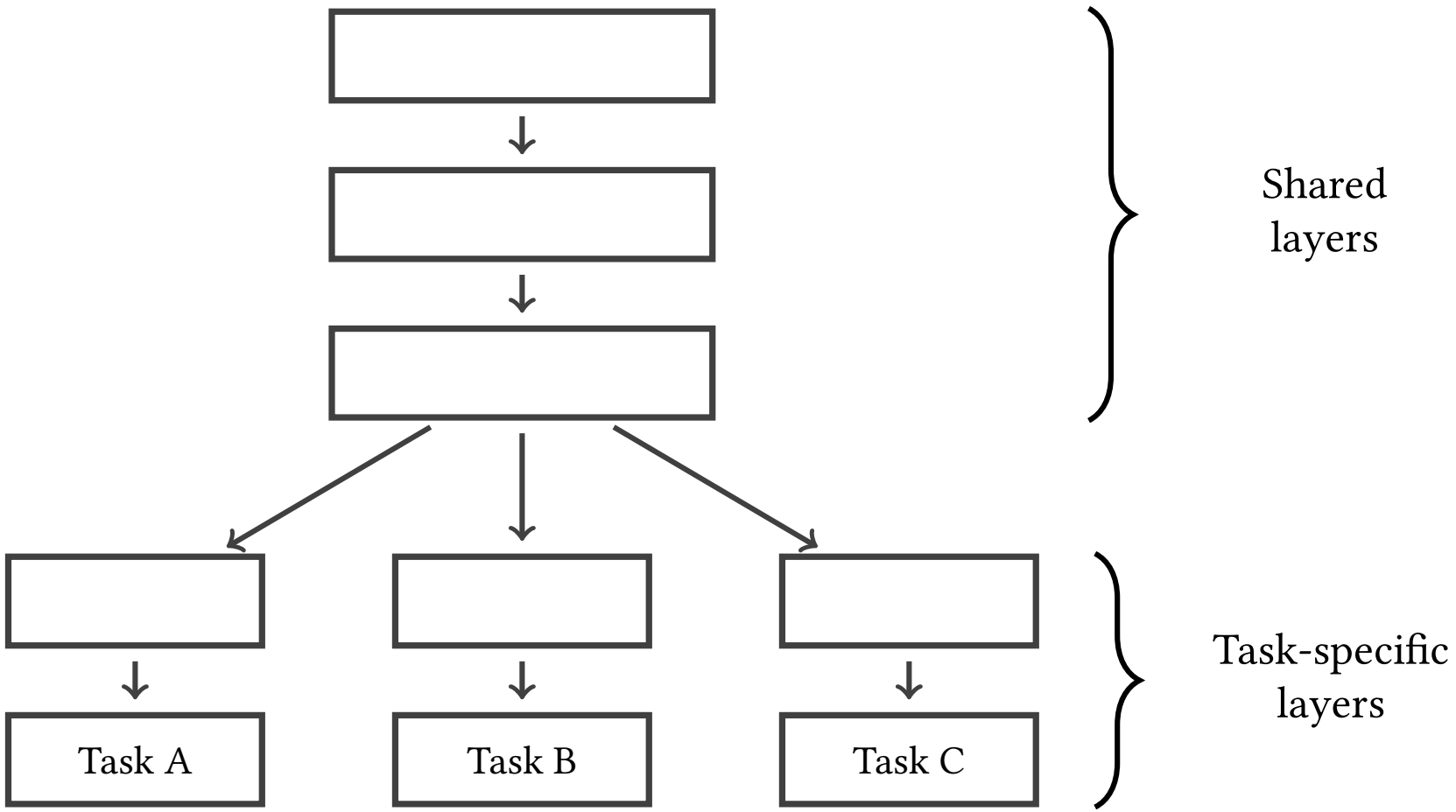


Arquitectura: compartición suave de parámetros



Yang & Hospedales. *Trace Norm Regularised Deep Multi-Task Learning*, 2017.

Arquitectura: compartición dura de parámetros



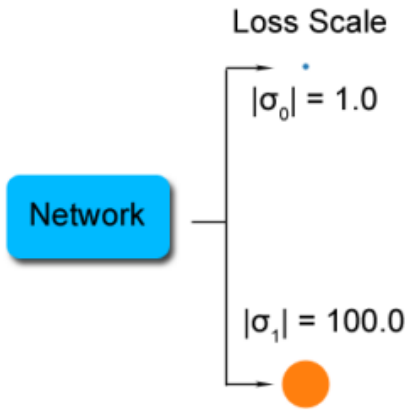
Curana. *Multitask learning: A knowledge-based source of inductive bias.* 1993.

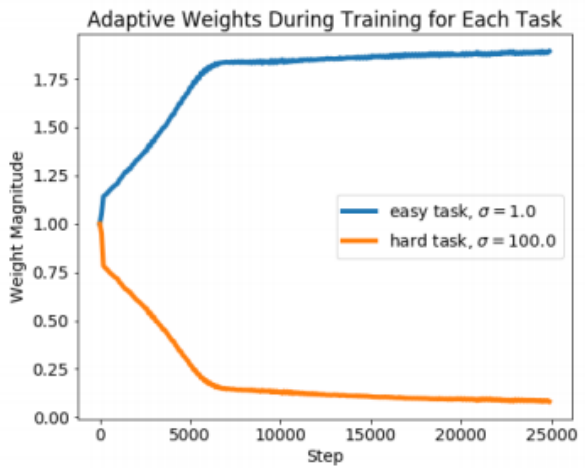
¡tiempo de codear!

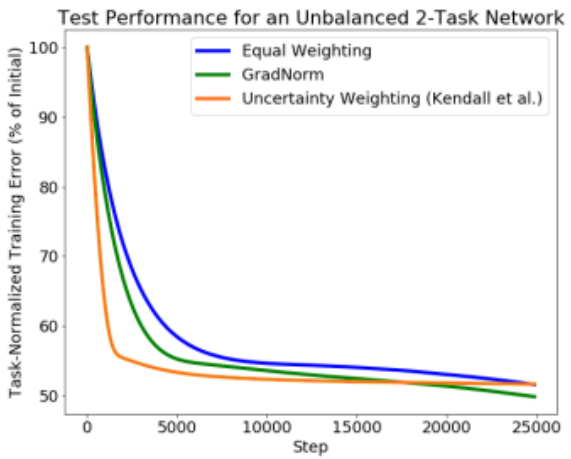
5_multitarea.ipynb

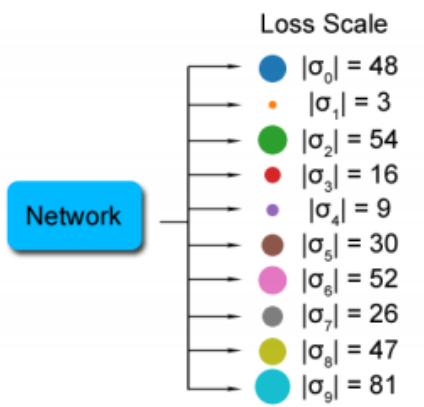
Problemas abiertos: convergencia

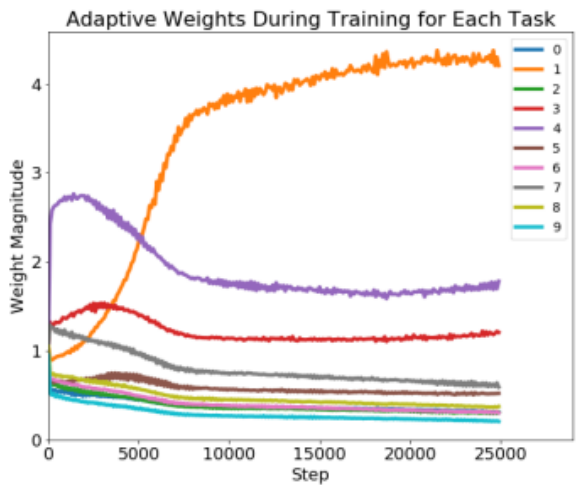
GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks

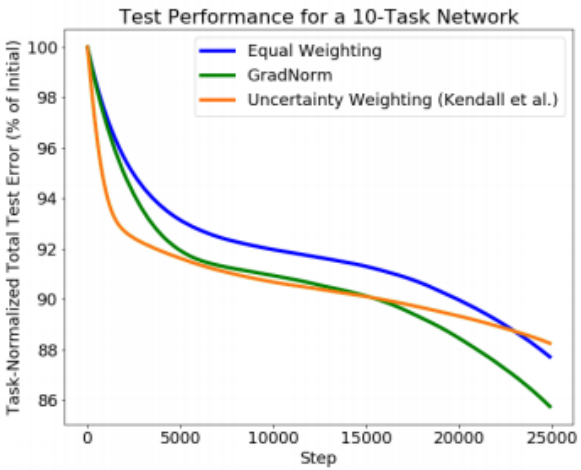






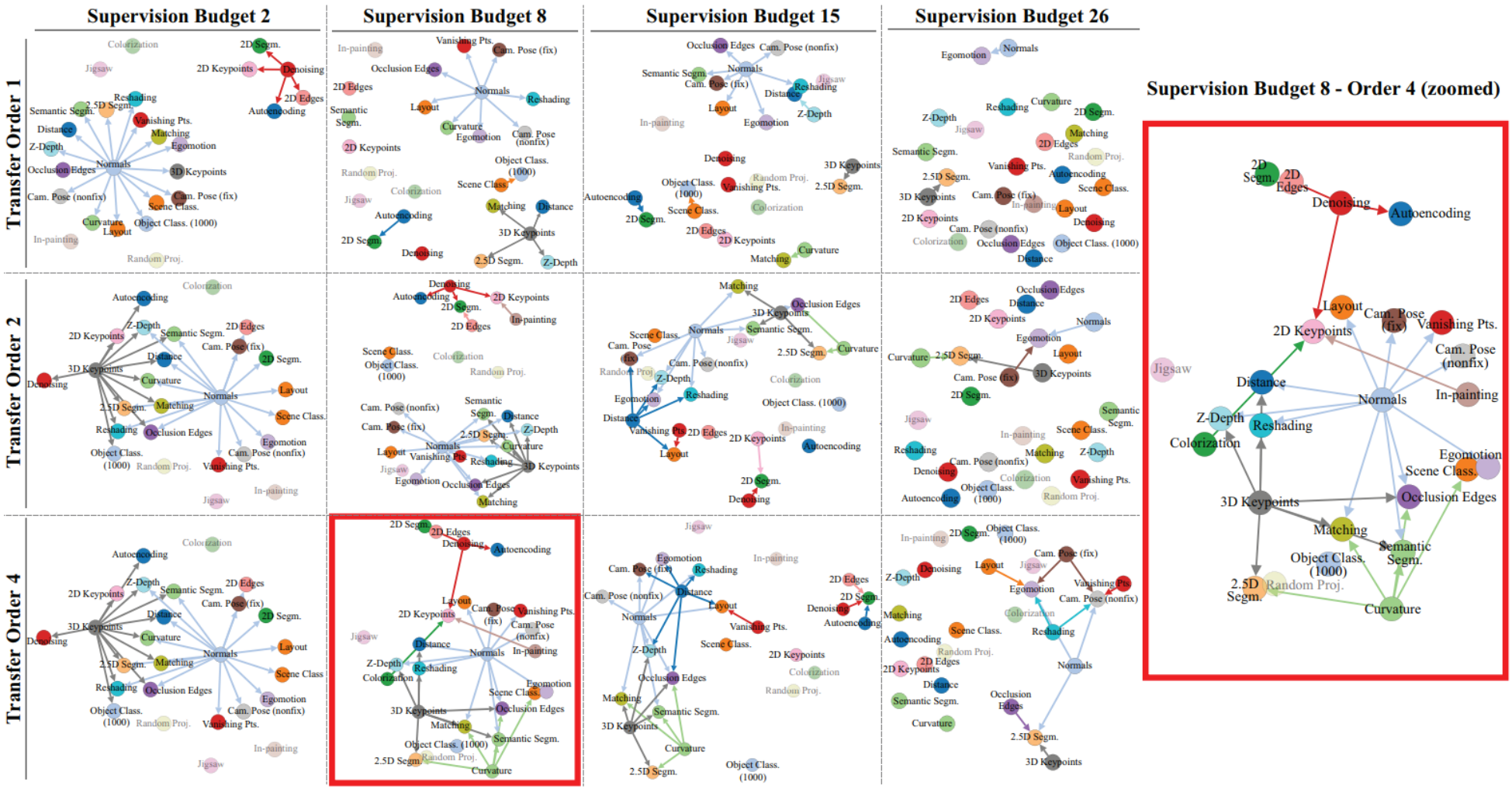






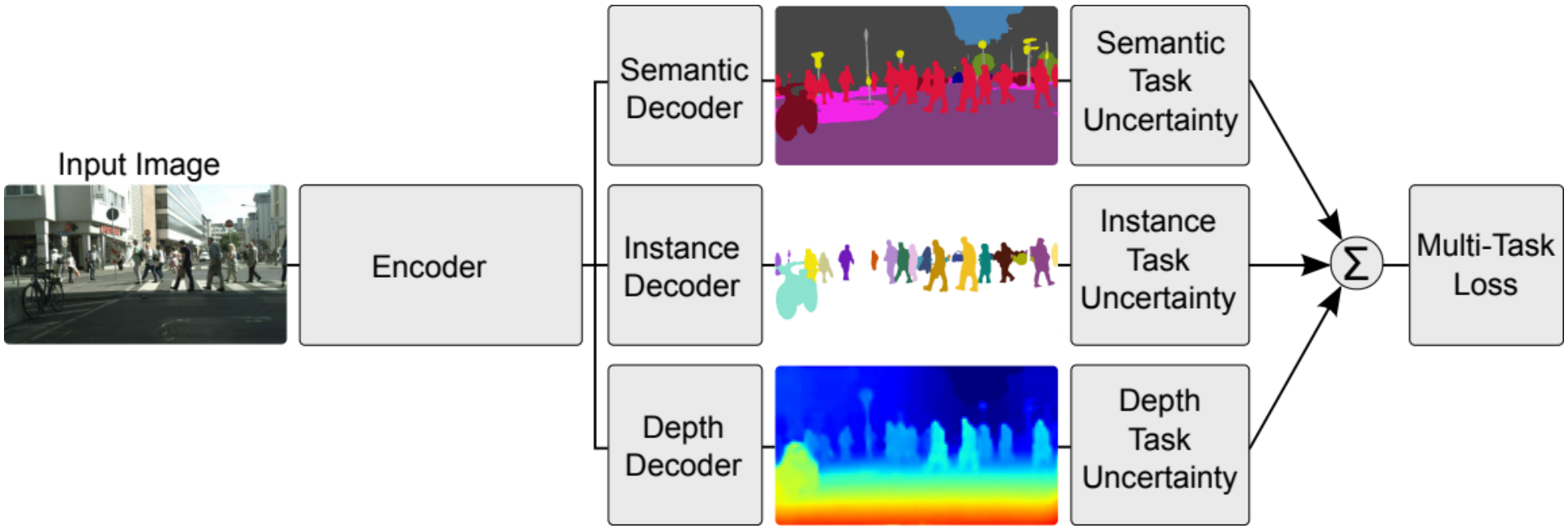
Chen et al. *GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks.* 2018.

Problemas abiertos: similitud de las tareas



A. Zamir et al. *Taskonomy: Disentangling Task Transfer Learning*. 2018.

Problemas abiertos: representaciones



Kendall et al. *Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics.* 2017.

Consejos para explorar multitarea en tu problema

- Obten un conjunto con varios objetivos o varios conjuntos monotarea.
- Explora y razona intuitivamente sobre la relación de las tareas.
- Entrena modelos sencillos para cada tarea.
- Comienza entrenando modelos multitarea para las tareas más sencillas y relacionadas.
- Verifica que las perdidas estén en la misma escala.
- Prueba optimizadores independientes.
- Suerte!

¡Gracias!