Introducción a la Detección de Objetos

Agustín Azzinnari (@ganitsu)

tryo·labs

FACULTAD DE INGENIERIA

¿Qué puedo hacer con una imagen?

(millones de sumas
y multiplicaciones
después…)

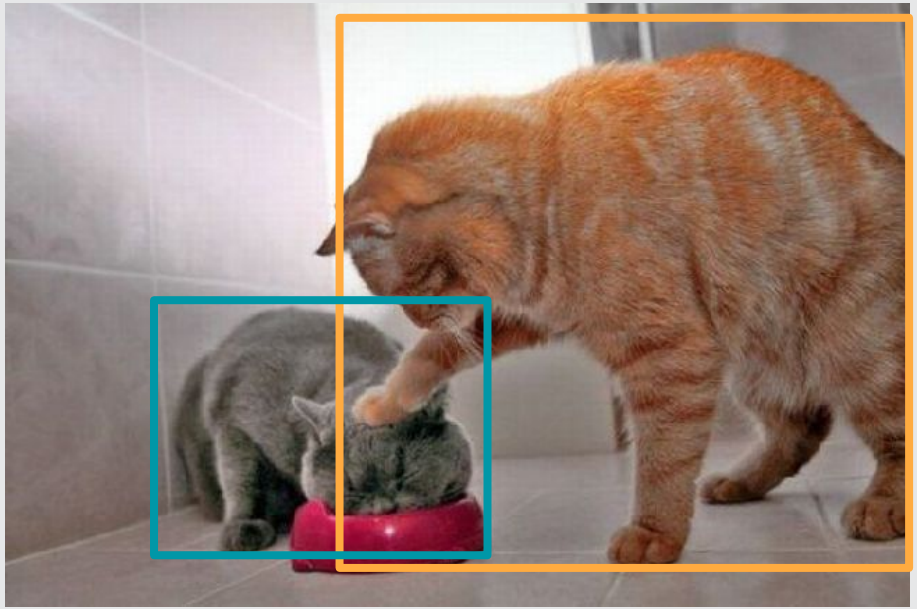
(y muchos kWh
gastados después...)

(y mucho tiempo
después...)

Hay un gato.

¿Alcanza con que me digan que hay un gato?

De clasificación a detección



Clasificación

Hay un gato en la foto

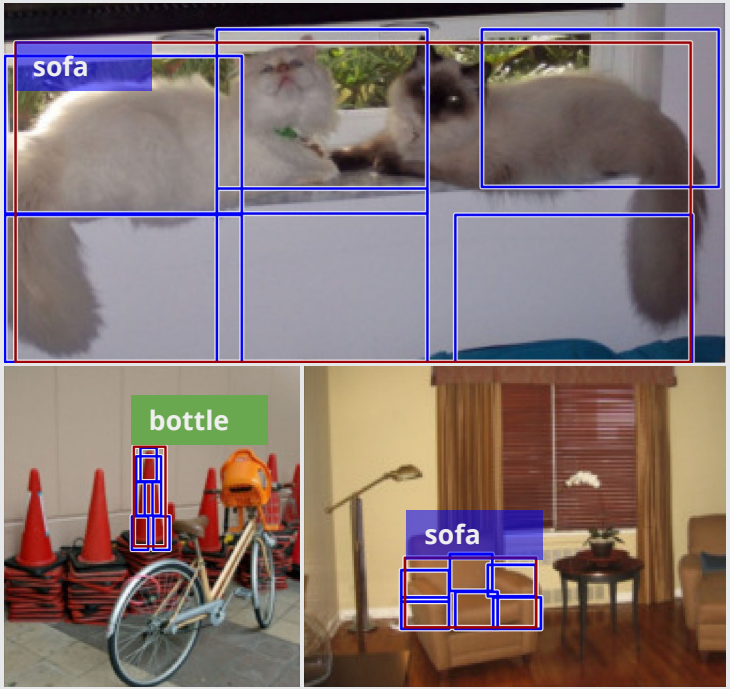
Localización

Hay un gato en la foto y está acá

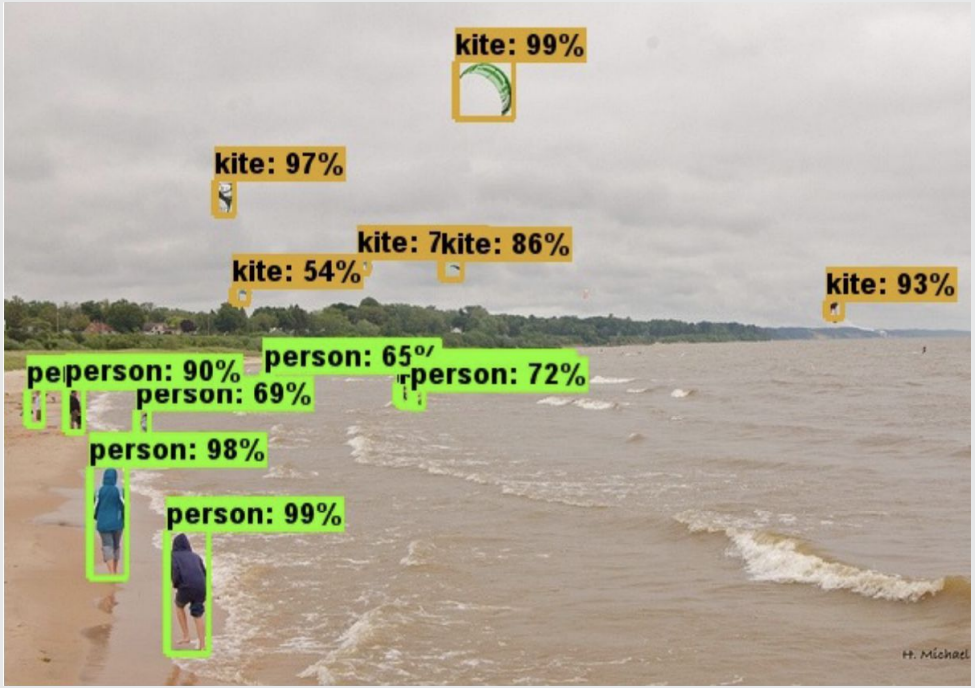
Detección

Hay dos gatos en la foto y están acá y acá

Antes vs ahora



Felzenszwalb et. al., "Object Detection with Discriminatively Trained Part Based Models", IEEE Transactions on Pattern Analysis and Machine Intelligence, 32, 2010.



Detected objects in a sample image (from the COCO dataset) (2017). Source: Google Research Blog.

¿Qué ve una computadora?

Una matriz muy muy grande.
Millones de entradas.

$$\begin{pmatrix} \begin{pmatrix} 251 & 32 & \cdots & 85 & 72 \\ 211 & 12 & \cdots & 125 & 131 \\ \begin{pmatrix} 251 & 32 & \cdots & 85 & 72 \\ 17 & 232 & & 91 & 90 \\ \vdots & & \ddots & & \vdots \\ 114 & 25 & & 12 & 14 \\ 172 & 51 & \cdots & 13 & 20 \end{pmatrix} \end{pmatrix} \end{pmatrix}$$

720

1280

rojo
verde
azul

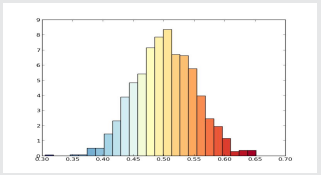
Desafíos en la detección de objetos

Aprendizaje automático

Modelo Estadístico

Clásico

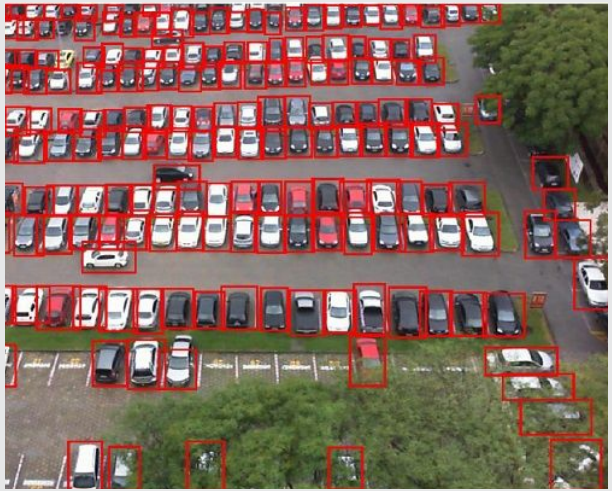
Extraigo **features** de las imágenes y las paso por un algoritmo de clasificación.



Deep learning

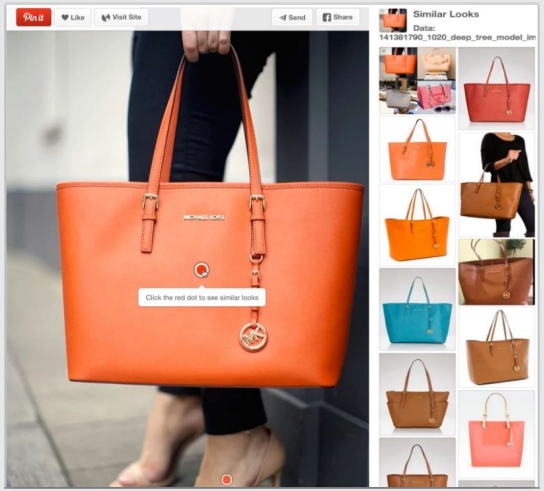
Paso la imagen **directamente** por un algoritmo de clasificación **más complejo**.

Aplicaciones de la detección de objetos



Hsieh et. al., “Drone-based Object Counting by Spatially Regularized Regional Proposal Networks”, ICCV 2017.

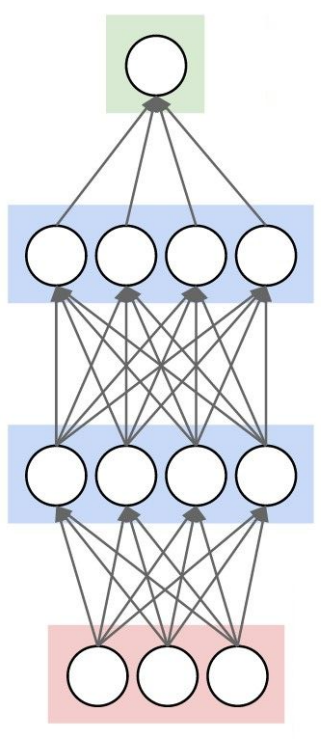
CT scan of a lung cancer patient at the Jingdong Zhongmei private hospital in Yanjiao, China's Hebei Province (AP Photo/Andy Wong)



Source: Pinterest

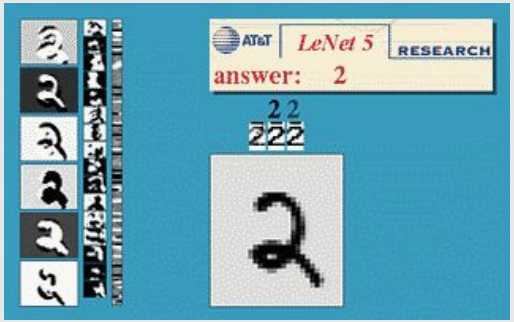
Deep Learning

Redes neuronales



Redes neuronales para clasificación, surge en los 80.

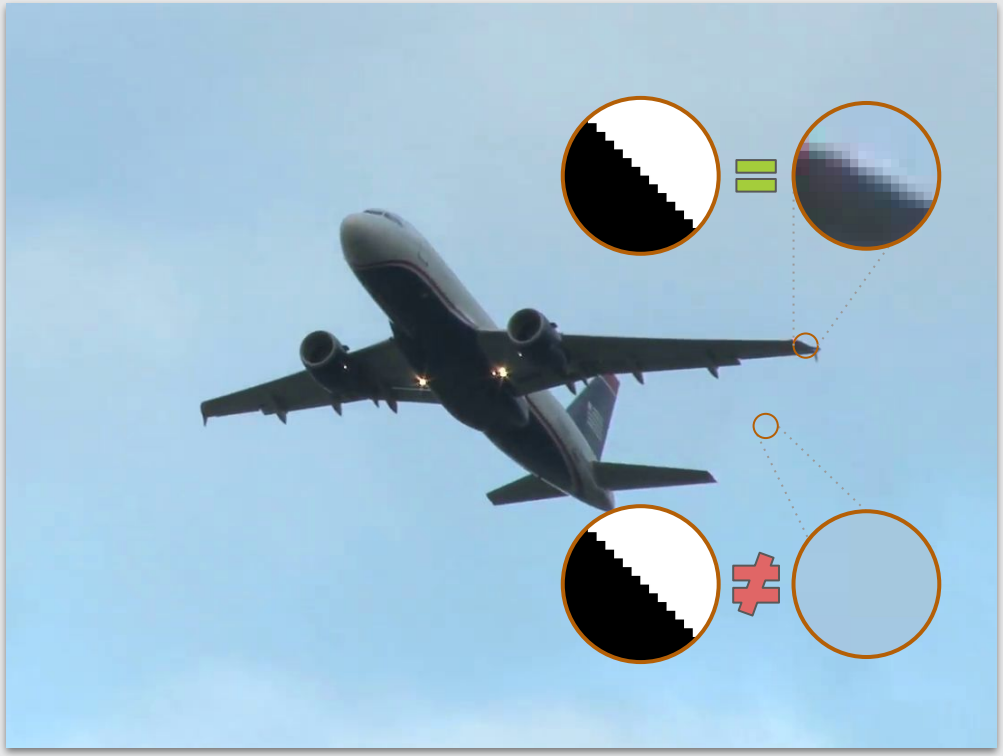
Utilización de **convoluciones** en estas redes (Yann LeCun, 1989), para crear **redes neuronales convolucionales**.



Lector de dígitos en cheques.

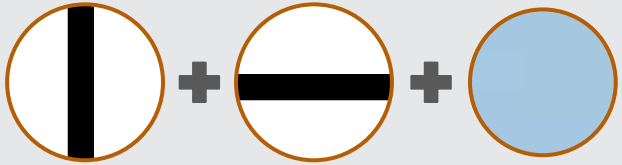
LeNet-5, Yann LeCun, 1998.

Convoluciones



Filtro que mira una **región pequeña** de la imagen y detecta un determinado patrón.

Los **encadeno** y detecto patrones más **complejos**.



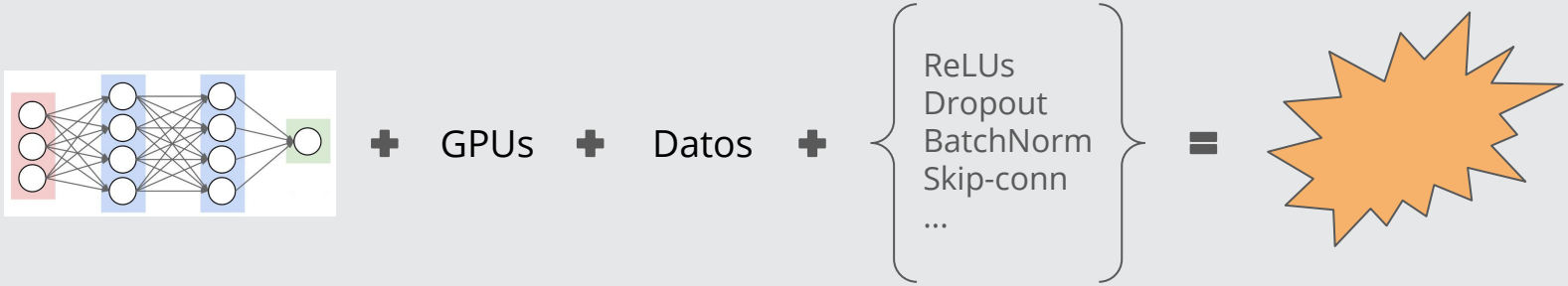
Preguntas

¿Tengo que configurar estos filtros a mano?

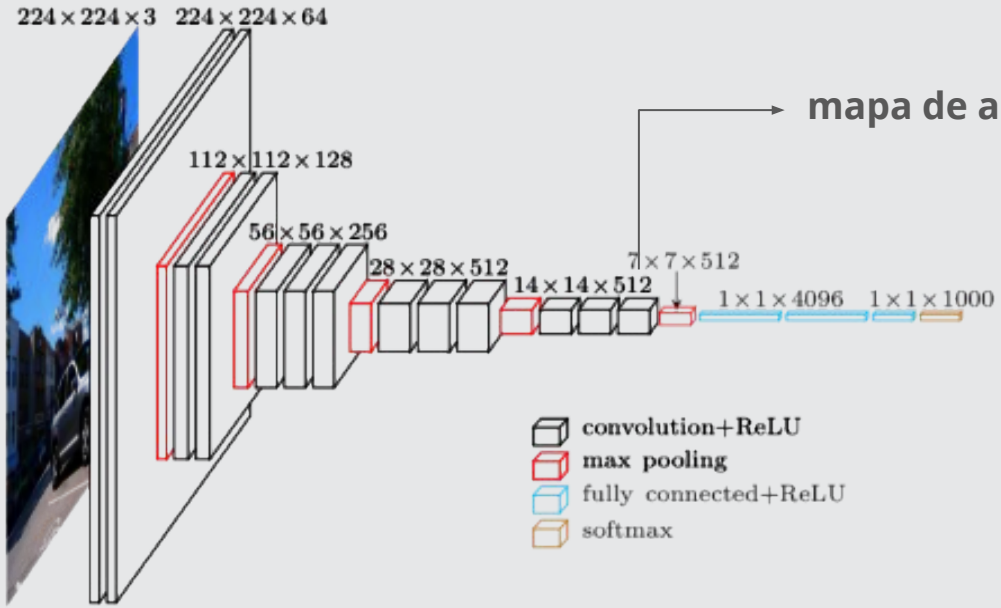
No. Se infieren los parámetros mirando muchas imágenes.

Entrenamiento mediante **descenso de gradiente**.

¿Por qué ahora?



Visualizando una red convolucional



Pre-entrenado:

IMAGENET

Figure from https://blog.heuritech.com/2016/02/29/a-brief-report-of-the-heuritech-deep-learning-meetup-5/

¿Y la detección?

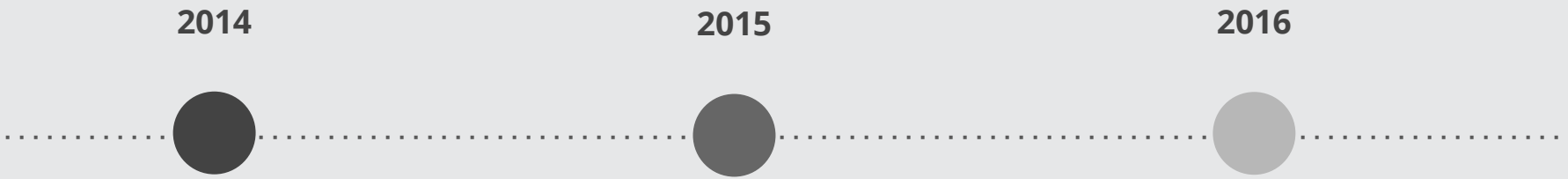
Qué está faltando:

- Estoy tratando con imágenes de **tamaño fijo**.
- Me detecta **un único objeto** (si hay más de uno, elige uno).
- No me da una **ubicación**.

Detección de Objetos: Faster R-CNN

Historia

Evolución de métodos propuestos a lo largo de tres años:

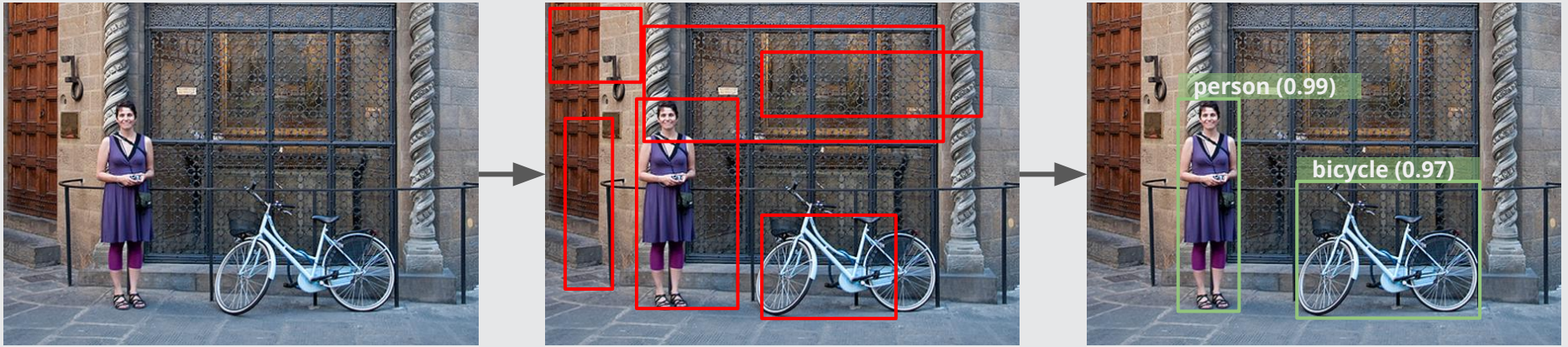


Ren, Girshick et. al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", CVPR 2016.

Arquitectura

Faster R-CNN es un **detector de objetos en dos etapas**:

- **Proposición de regiones interesantes**. ¿Dónde vale la pena mirar?
- **Análisis de regiones**. ¿Hay algo en esas regiones?



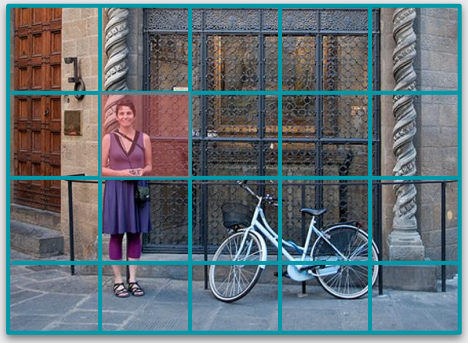
Proponiendo regiones (I)

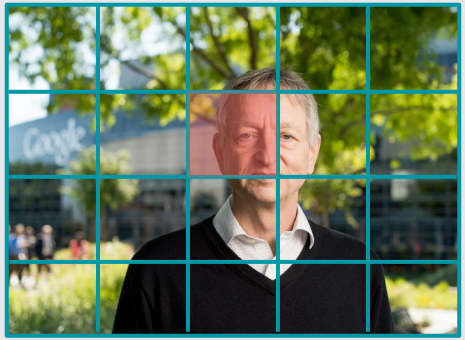
¿Qué quiero? Regiones de interés, donde puede haber algo.

¿Con qué cuento? Mi red convolucional, con sus filtros y su **mapa de activaciones**.

Proponiendo regiones (II)

También me puede ayudar con el **tamaño**:



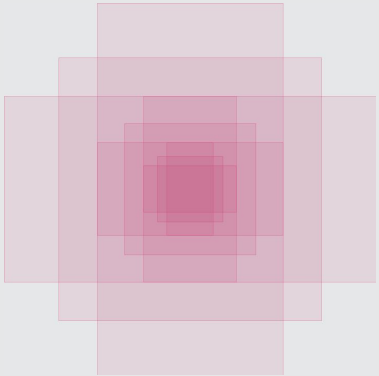


Uso **dos** clasificadores:

- Uno que me diga **si parece haber un objeto**.
- Otro que me diga **qué tamaño tendría la caja**.

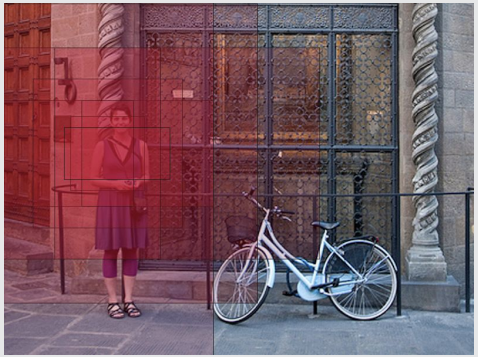
Proponiendo regiones (III)

Mi imagen puede tener cualquier tamaño. Uso **cajas de referencia** (o **anchors**) para modelar su ubicación:



Cajas de referencia

Posiciones del mapa de activaciones



Cajas de referencia en **un punto** particular

Usando las regiones (I)

Consideraciones:

- Todas estas regiones tienen **tamaños distintos**.
- Las activaciones salen de la **región original** (antes de agrandar).

¿Se **activan nuevos filtros** en la región agrandada?

Vinculo **nueva región** con el **mapa de activaciones**. Y llevo a tamaño único.

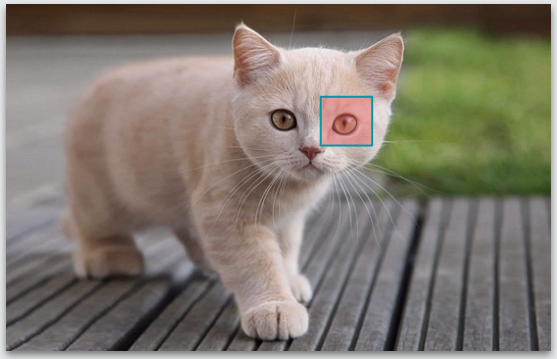
Ahora sí tengo una **descripción adecuada** de mi región.

Usando las regiones (II)

Paso por dos nuevos clasificadores:

- **¿Qué objeto es mi región?** O clase *background*.
- **¿Cómo tengo que cambiar el tamaño la región?**

Esto último lo hacemos por clase:



Resumen

En definitiva:

- Utilizo **mapa de activaciones** como base.
- **Propongo regiones** en una primera etapa.
- Utilizo y **refino regiones** en una segunda.

Me quedo con las **regiones que detectan objetos** (buen puntaje).

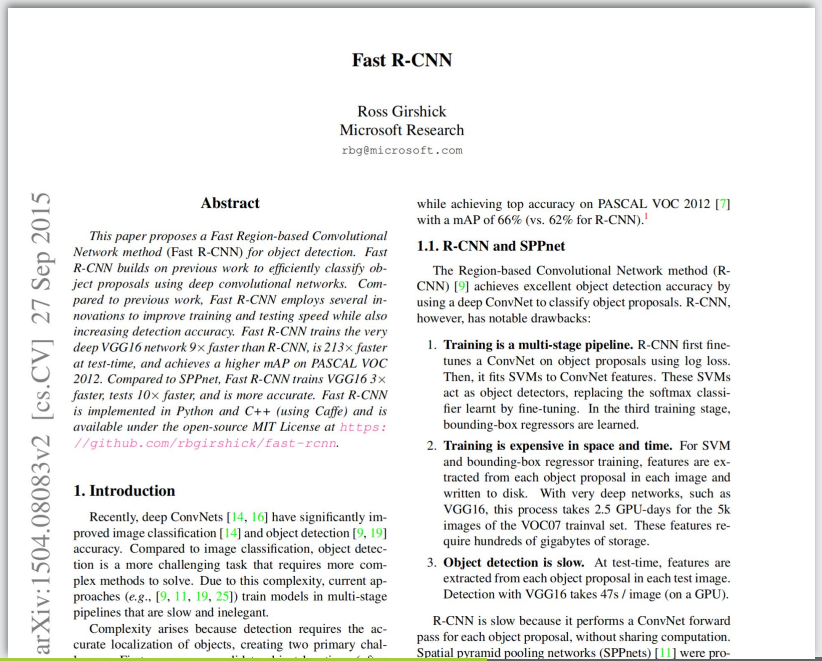
¿Cómo se ve?

```
92          # Decode boxes
93          all_proposals = decode(all_anchors, rpn_bbox_pred)
94
95          # Filter proposals with less than threshold probability.
96          min_prob_filter = tf.greater_equal(
97              all_scores, self._min_prob_threshold
98          )
99
100         # Filter proposals with negative or zero area.
101         (x_min, y_min, x_max, y_max) = tf.unstack(all_proposals, axis=1)
102         zero_area_filter = tf.greater(
103             tf.maximum(x_max - x_min, 0.0) * tf.maximum(y_max - y_min, 0.0),
104             0.0
105         )
106         proposal_filter = tf.logical_and(zero_area_filter, min_prob_filter)
107
108         # Filter proposals and scores.
109         all_proposals_total = tf.shape(all_scores)[0]
110         unsorted_scores = tf.boolean_mask(
111             all_scores, proposal_filter,
112             name='filtered_scores'
113         )
```

Desafíos

Implementar de un paper

¿Cómo se publican los algoritmos?
¡Todo texto, y manejate!



Fast R-CNN

Ross Girshick
Microsoft Research
rbg@microsoft.com

arXiv:1504.08083v2 [cs.CV] 27 Sep 2015

Abstract

This paper proposes a Fast Region-based Convolutional Network method (Fast R-CNN) *for object detection. Fast R-CNN builds on previous work to efficiently classify object proposals using deep convolutional networks. Compared to previous work, Fast R-CNN employs several innovations to improve training and testing speed while also increasing detection accuracy. Fast R-CNN trains the very deep VGG16 network 9× faster than R-CNN, is 213× faster at test-time, and achieves a higher mAP on PASCAL VOC 2012. Compared to SPPnet, Fast R-CNN trains VGG16 3× faster, tests 10× faster, and is more accurate. Fast R-CNN is implemented in Python and C++ (using Caffe) and is available under the open-source MIT License at* https://github.com/rbgirshick/fast-rcnn.

1. Introduction

Recently, deep ConvNets [14, 16] have significantly improved image classification [14] and object detection [9, 19] accuracy. Compared to image classification, object detection is a more challenging task that requires more complex methods to solve. Due to this complexity, current approaches (*e.g.*, [9, 11, 19, 25]) train models in multi-stage pipelines that are slow and inelegant.

Complexity arises because detection requires the accurate localization of objects, creating two primary chal-

while achieving top accuracy on PASCAL VOC 2012 [7] with a mAP of 66% (vs. 62% for R-CNN).[1]

1.1. R-CNN and SPPnet

The Region-based Convolutional Network method (R-CNN) [9] achieves excellent object detection accuracy by using a deep ConvNet to classify object proposals. R-CNN, however, has notable drawbacks:

1. **Training is a multi-stage pipeline.** R-CNN first fine-tunes a ConvNet on object proposals using log loss. Then, it fits SVMs to ConvNet features. These SVMs act as object detectors, replacing the softmax classifier learnt by fine-tuning. In the third training stage, bounding-box regressors are learned.
2. **Training is expensive in space and time.** For SVM and bounding-box regressor training, features are extracted from each object proposal in each image and written to disk. With very deep networks, such as VGG16, this process takes 2.5 GPU-days for the 5k images of the VOC07 trainval set. These features require hundreds of gigabytes of storage.
3. **Object detection is slow.** At test-time, features are extracted from each object proposal in each test image. Detection with VGG16 takes 47s / image (on a GPU).

R-CNN is slow because it performs a ConvNet forward pass for each object proposal, without sharing computation. Spatial pyramid pooling networks (SPPnets) [11] were pro-

2.1. The RoI pooling layer

The RoI pooling layer uses max pooling to convert the features inside any valid region of interest into a small feature map with a fixed spatial extent of $H \times W$ (*e.g.*, 7×7), where H and W are layer hyper-parameters that are independent of any particular RoI. In this paper, an RoI is a rectangular window into a conv feature map. Each RoI is defined by a four-tuple (r, c, h, w) that specifies its top-left corner (r, c) and its height and width (h, w).

RoI max pooling works by dividing the $h \times w$ RoI window into an $H \times W$ grid of sub-windows of approximate size $h/H \times w/W$ and then max-pooling the values in each sub-window into the corresponding output grid cell. Pooling is applied independently to each feature map channel, as in standard max pooling. The RoI layer is simply the special-case of the spatial pyramid pooling layer used in SPPnets [11] in which there is only one pyramid level. We use the pooling sub-window calculation given in [11].

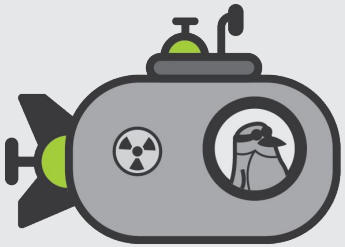
Desafíos al implementar un paper

Detalles de implementación no tienen lugar en papers académicos

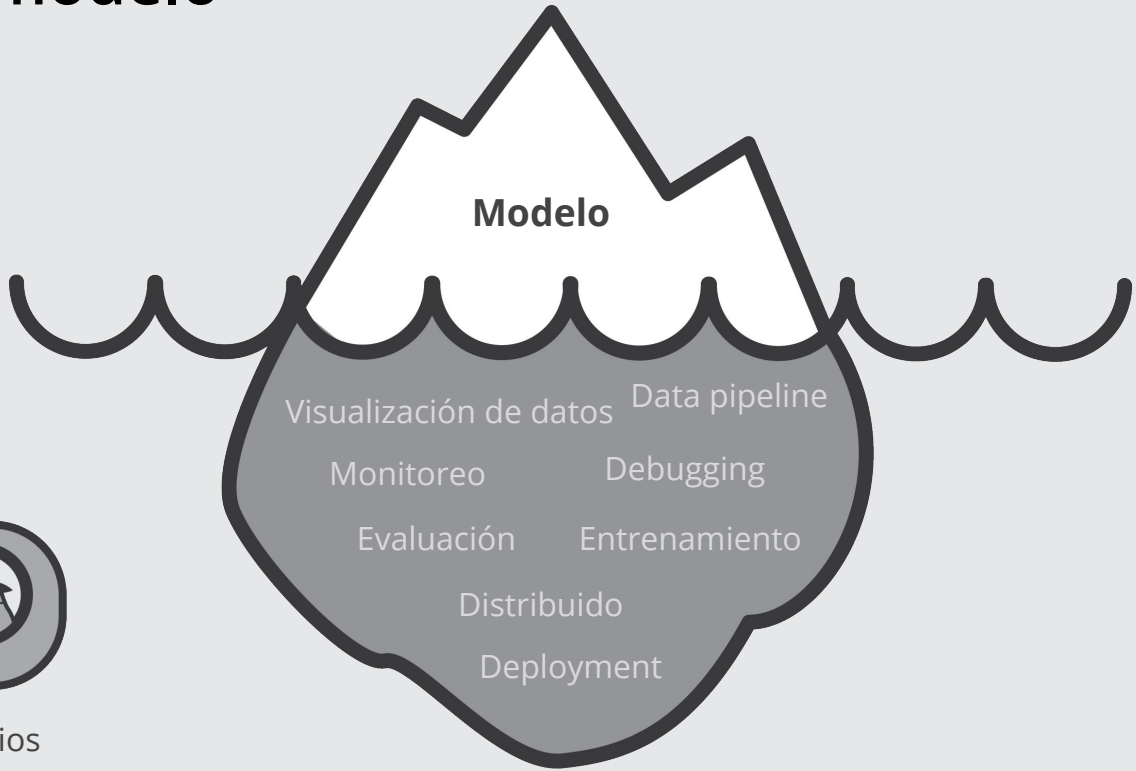
Los papers están congelados en el tiempo

Hay muchas formas de implementarlos

Más allá del modelo



Tests unitarios

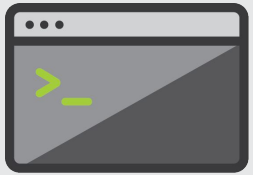


Luminoth

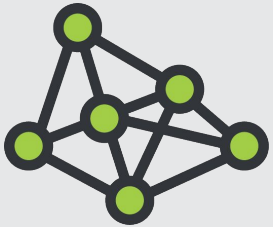
¿Qué es Luminoth?

LUMINOTH

Librería/herramienta open-source para ~~visión por computadora~~ detección de objetos.



CLI
tools



Modelos
pre-entrenados

Cloud
integration

Objetivos

- Simple y pronto para usar
- Production ready
- Open source
- Código legible
- Extensible y modular

Simplicidad como objetivo

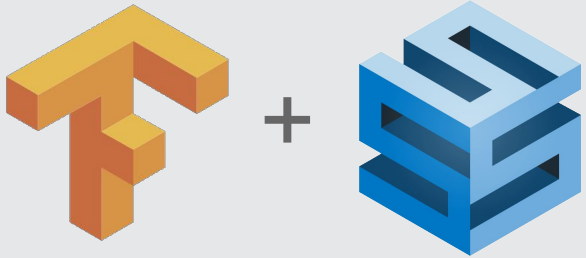
LUMINOTH

Shell

```
$ pip install luminoth
$ lumi predict video.mp4 -k car
Found 1 files to predict.
Neither checkpoint not config
specified, assuming `accurate`.
Predicting video.mp4
    [##############] 100% fps: 5.9
```

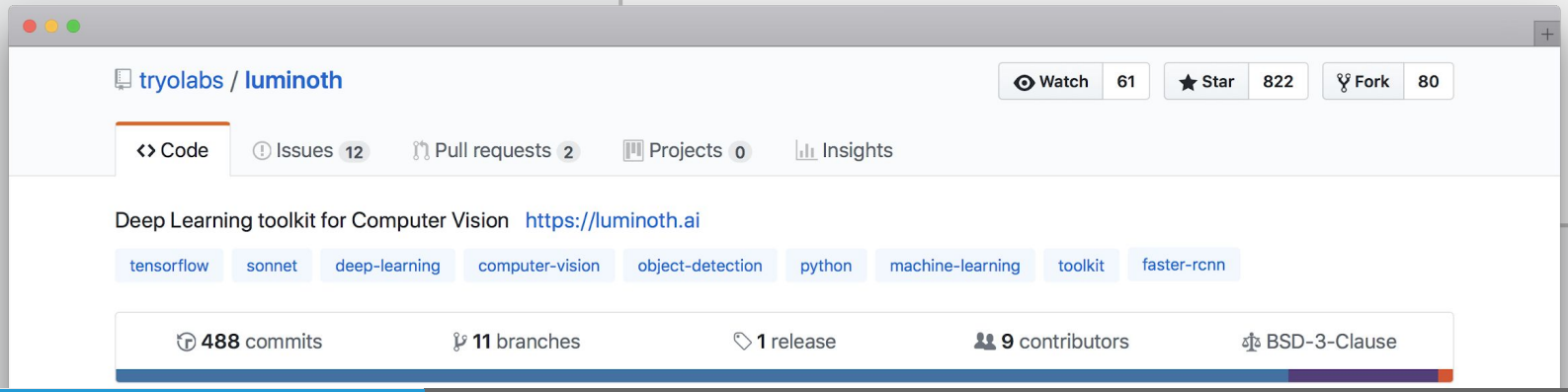
Código abierto

LUMINOTH



```
import sonnet as snt

def RPN(snt.AbstractModule):
    def __init__(self, *args, name='rpn'):
        [...]  # submodules init, config
```



Usando Luminoth

LUMINOTH

https://github.com/tryolabs/luminoth

Shell

```
$ pip install luminoth
$ lumi --help
Usage: lumi [OPTIONS] COMMAND [ARGS]...

Options:
  -h, --help  Show this message and exit.
Commands:
  checkpoint  Groups of commands to manage checkpoints
  cloud       Groups of commands to train models in the cloud
  dataset     Groups of commands to manage datasets
  eval        Evaluate trained (or training) models
  predict     Obtain a model's predictions
  server      Groups of commands to serve models
  train       Train models
```

Ciclo de uso de Luminoth

LUMINOTH

Shell

```
$ lumi dataset transform --type pascal --data-dir /data/pascal --output /data/
# Create tfrecords for optimizing data consumption.
$ lumi train --config pascal-fasterrcnn.yml
# Hours of training...

$ tensorboard --logdir jobs/

# On another GPU/Machine/CPU.
$ lumi eval --config pascal-fasterrcnn.yml
# Checks for new checkpoints and writes logs.

# Finally.
$ lumi server web --config pascal-fasterrcnn.yml
# Looks for checkpoint and loads it into a simple frontend/json API server.
```

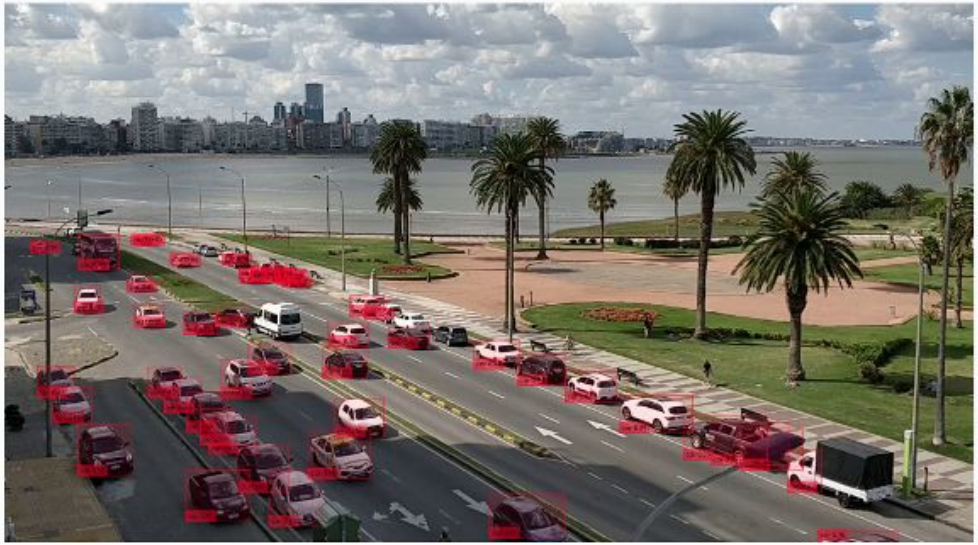
LUMINOTH

Image Browse... traffic.png Submit

Probability threshold 0.48

Raw API response:


```
{
  "objects": [
    {
      "bbox": [
        144,
        844,
        757,
        948
      ],
      "label": "car",
      "prob": 0.9883
    },
    {
      "bbox": [
        359,
        935,
        487,
        1039
      ],
      "label": "car",
      "prob": 0.974
    },
    {
      "bbox": [
        641,
        638,
        722,
        688
      ],
      "label": "car",
      "prob": 0.9693
    },
```

tryo·labs

Muchas gracias por escuchar! Preguntas?

Agustín Azzinnari

@ganitsu

@tryolabs

github.com/tryolabs/luminoth