Object detection with CNNs

and how it is related to the analysis of charts

Tiago da Silva

November 12, 2021

Escola de Matemática Aplicada (FGV, Rio de Janeiro)

Table of contents

Introduction

Object Detection

The classification of images according to a set of labels is a common task for CNNs; nevertheless, there are situations in which a distinct regions of an image could be assigned to distinct labels. The Object Detection methods, then, aim to address those scenarios.

Object Detection

The classification of images according to a set of labels is a common task for CNNs; nevertheless, there are situations in which a distinct regions of an image could be assigned to distinct labels. The Object Detection methods, then, aim to address those scenarios.

Cat

Figure 1: Image classification.

Object Detection

The classification of images according to a set of labels is a common task for CNNs; nevertheless, there are situations in which a distinct regions of an image could be assigned to distinct labels. The Object Detection methods, then, aim to address those scenarios.

Human, Cat

Figure 1: Object detection.

Cat

Figure 2: Image classification.

Problem

Let $A \in \mathbb{R}^{m \times n}$ be a matrix with m rows and n columns and $\mathcal{L} \subset \mathbb{N}$ be a (finite) set of labels. In this context, our goal is to search all $b \in I_m^2 \times I_n^2$, $b = (x_1, x_2, y_1, y_2)$[1], such that the matrix

$$B = A[x_1 : x_2, y_1 : y_2]$$

could be assigned to a label $i \in \mathcal{L}$; the vector b is the *bounding box* of the object B. Explicitly, our method should compute a list of vectors of the form

$$(x_1, x_2, y_1, y_2, i),$$

in which $(x_1, x_2) \in I_m^2$, $(y_1, y_2) \in I_n^2$ and $i \in \mathcal{L}$.

[1]We write $I_n = \{u \in \mathbb{N} \cup \{0\} : u < n\}$.

Object Detection

In this context, there are some questions that must me conveniently addressed; for instance,

1. How can we make proposals for the bounding boxes?

Object Detection

In this context, there are some questions that must me conveniently addressed; for instance,

1. How can we make proposals for the bounding boxes?
   Given an appropriate set of boxes, we could use standard image classifications methods to label them; the size of this set, however, should ensure the computational feasibility of this task.

Object Detection

In this context, there are some questions that must me conveniently addressed; for instance,

1. How can we make proposals for the bounding boxes?
   Given an appropriate set of boxes, we could use standard image classifications methods to label them; the size of this set, however, should ensure the computational feasibility of this task.
2. What is an convenient way of measuring loss?

Object Detection

In this context, there are some questions that must me conveniently addressed; for instance,

1. How can we make proposals for the bounding boxes?
   Given an appropriate set of boxes, we could use standard image classifications methods to label them; the size of this set, however, should ensure the computational feasibility of this task.
2. What is an convenient way of measuring loss?
   We must design a multi-task system: on the one hand, it performs a classification of the boxes; on the other hand, it performs a regression task to compute those boxes.

Object Detection

In this context, there are some questions that must me conveniently addressed; for instance,

1. How can we make proposals for the bounding boxes?
   Given an appropriate set of boxes, we could use standard image classifications methods to label them; the size of this set, however, should ensure the computational feasibility of this task.
2. What is an convenient way of measuring loss?
   We must design a multi-task system: on the one hand, it performs a classification of the boxes; on the other hand, it performs a regression task to compute those boxes.

In this presentation, I will (auspiciously) provide satisfactory (yet definetely not exhaustive) directions to remedy these situations.

CNNs?

Since 2013, deep CNNs were introduced as a state-of-the-art procedure for object detection. The next slides, then, introduce popular architectures (apart from their age, they are still used nowadays[2]).

1. RCNN (2014) [4]

[2]RCNN isn't; nevertheless, it was the spark that lightened the fire for more robust procedures, so we must include it.

CNNs?

Since 2013, deep CNNs were introduced as a state-of-the-art procedure for object detection. The next slides, then, introduce popular architectures (apart from their age, they are still used nowadays[2]).

1. RCNN (2014) [4]
2. Fast RCNN (2015) [3]

[2] RCNN isn't; nevertheless, it was the spark that lightened the fire for more robust procedures, so we must include it.

CNNs?

Since 2013, deep CNNs were introduced as a state-of-the-art procedure for object detection. The next slides, then, introduce popular architectures (apart from their age, they are still used nowadays[2]).

1. RCNN (2014) [4]
2. Fast RCNN (2015) [3]
3. Faster RCNN (2016) [11]

[2] RCNN isn't; nevertheless, it was the spark that lightened the fire for more robust procedures, so we must include it.

CNNs?

Since 2013, deep CNNs were introduced as a state-of-the-art procedure for object detection. The next slides, then, introduce popular architectures (apart from their age, they are still used nowadays[2]).

1. RCNN (2014) [4]
2. Fast RCNN (2015) [3]
3. Faster RCNN (2016) [11]
4. Mask RCNN (2017) [5]

[2]RCNN isn't; nevertheless, it was the spark that lightened the fire for more robust procedures, so we must include it.

CNNs?

Since 2013, deep CNNs were introduced as a state-of-the-art procedure for object detection. The next slides, then, introduce popular architectures (apart from their age, they are still used nowadays[2]).

1. RCNN (2014) [4] $\leftarrow$
2. Fast RCNN (2015) [3] $\leftarrow$
3. Faster RCNN (2016) [11] $\leftarrow$
4. Mask RCNN (2017) [5]

[2] RCNN isn't; nevertheless, it was the spark that lightened the fire for more robust procedures, so we must include it.

RCNN

The RCNN ("Regions with CNN") was proposed as a three step system for object detection and classification: (1) region proposals, (2) feature extraction and (3) classification.

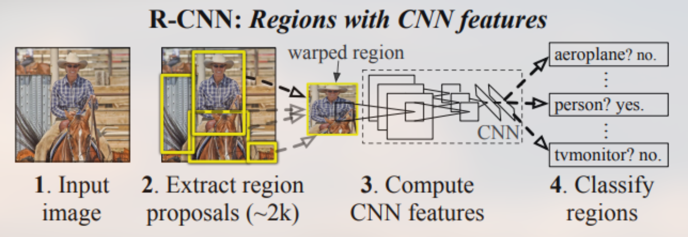


Figure 3: System proposed in [4].

RCNN: Region proposals

The region proposal method is, actually, independent and, to some extent, it is innocuous to the subsequent steps; originally, the **Selective Search** [13], a greedy algortihm aided with a graph-based[3] image segmentation algorithm, was used, so we describe it briefly.

[3]Precisely, we must design a undirected weighted graph $G = (V, E)$ with vertices V, the pixels, and edges E; the weights are a measure of dissimilarity between the pixels. Heuristically, a region is a connected component such that the discrepancy between the internal dissimilarity and the dissimilarity with its neighbours are enoughly distinguishable.

RCNN: Region proposals

The region proposal method is, actually, independent and, to some extent, it is innocuous to the subsequent steps; originally, the **Selective Search** [13], a greedy algortihm aided with a graph-based image segmentation algorithm, was used, so we describe it briefly.

(a) Regions

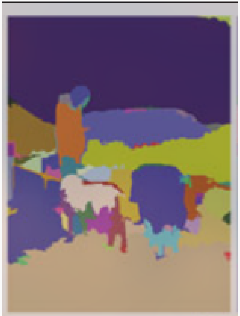
Initial proposals generated with a graph-based algorithm [2].

(b) Boxes

Figure 4: Selective search with graph-based image segmentation [13].

RCNN: Region proposals

The region proposal method is, actually, independent and, to some extent, it is innocuous to the subsequent steps; originally, the **Selective Search** [13], a greedy algortihm aided with a graph-based image segmentation algorithm, was used, so we describe it briefly.



(a) Regions

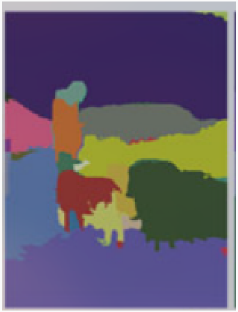
Compute similarities and merge boxes.

(b) Boxes

Figure 4: Selective search with graph-based image segmentation [13].

RCNN: Region proposals

The region proposal method is, actually, independent and, to some extent, it is innocuous to the subsequent steps; originally, the **Selective Search** [13], a greedy algortihm aided with a graph-based image segmentation algorithm, was used, so we describe it briefly.



(a) Regions

Reiterate.

(b) Boxes

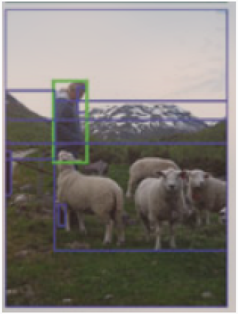
Figure 4: Selective search with graph-based image segmentation [13].

RCNN: Region proposals

The region proposal method is, actually, independent and, to some extent, it is innocuous to the subsequent steps; originally, the **Selective Search** [13], a greedy algortihm aided with a graph-based image segmentation algorithm, was used, so we describe it briefly.

(a) Regions

Region proposals.



(b) Boxes

Figure 4: Selective search with graph-based image segmentation [13].

RCNN: Feature extraction

We have the regions (around 2000 of them); the next task is the feature extraction. For this, the authors used the Caffe implementation of the CNN designed by Alex Krizhevsky[3] [7]; it extracts a feature vector in $\mathbb{R}^{4096}$.

[3] Currently, we call it "AlexNet".

RCNN: Feature extraction

RCNN: Region proposals and feature extraction [3].

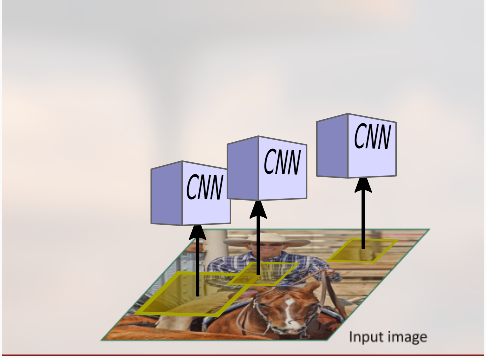
Initial image.

RCNN: Feature extraction

RCNN: Region proposals and feature extraction [3].

Regions of Interest (*RoI*), possibly computed with Selective Search.

RCNN: Feature extraction



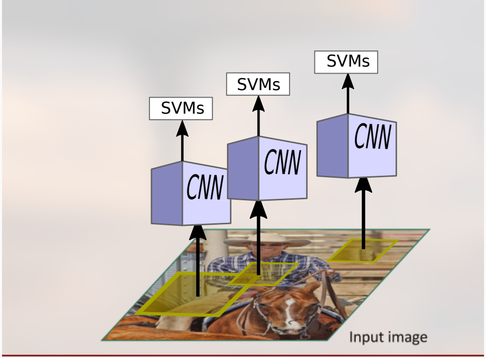
RCNN: Region proposals and feature extraction [3].

Feature extraction with AlexNet (also, the RoIs must be warped; the CNN requires a 227×227 image).

RCNN: Classification

Equipped with the feature vectors, we have all the machinery of classical machine learning at our disposal for class inference; the authors' choice was for a Support Vector Machine (SVM) model.

RCNN: Classification



RCNN: Classification [3].

Classification with SVMs.

RCNN: Summary

In summary, RCNN requires a independent method for region proposals; we could, for instance, use Selective Search, a heuristic approach implemented with a greedy algorithm. On the other hand, it separates the task of bounding box estimation and classification; then, each task is independently trained and uses a convenient loss function.

Fast RCNN

The multi-stage training in RCNN is computionally inappropriate: we must use distinct methods for training the CNN and the SVMs; also, we apply the forward pass of the CNN in each RoI, so we are not exploiting some computation shares. In this context, Fast RCNN proposes

(a) using the whole image as the network input, subsequently projecting the region proposals, which are still computed with **Selective Search** [13], in the network's output;

Fast RCNN

The multi-stage training in RCNN is computionally inappropriate: we must use distinct methods for training the CNN and the SVMs; also, we apply the forward pass of the CNN in each RoI, so we are not exploiting some computation shares. In this context, Fast RCNN proposes

(a) using the whole image as the network input, subsequently projecting the region proposals, which are still computed with **Selective Search** [13], in the network's output;

(b) fixing the offsets of the region proposals using a data-driven trainable procedure;

Fast RCNN

The multi-stage training in RCNN is computionally inappropriate: we must use distinct methods for training the CNN and the SVMs; also, we apply the forward pass of the CNN in each RoI, so we are not exploiting some computation shares. In this context, Fast RCNN proposes

(a) using the whole image as the network input, subsequently projecting the region proposals, which are still computed with **Selective Search** [13], in the network's output;

(b) fixing the offsets of the region proposals using a data-driven trainable procedure;

(c) and training a fully connected network that simutaneously compute the bounding boxes offsets and the class probabilities (with a softmax layer), allowing, therefore, a single-stage training.

Fast RCNN: Pipeline

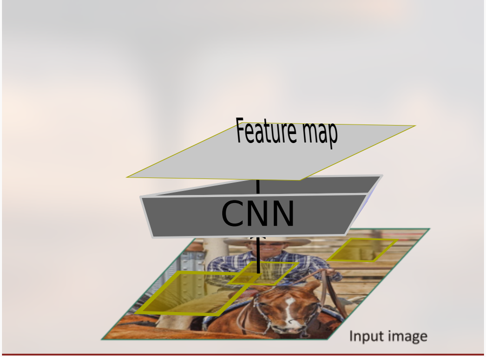
The region proposals are still a component of a kind of pre-processing stage; Fast RCNN, as a system, is designed to fix and to classify those boxes.

Fast RCNN: Pipeline

Fast RCNN: Pipeline [3].

Regions of Interest (Selective Search).

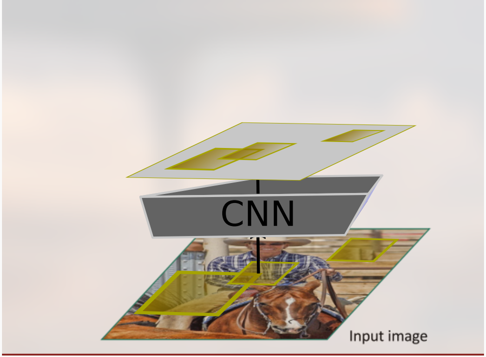
Fast RCNN: Pipeline



Fast RCNN: Pipeline [3].

Feature map computed in the `conv5` layer of AlexNet.

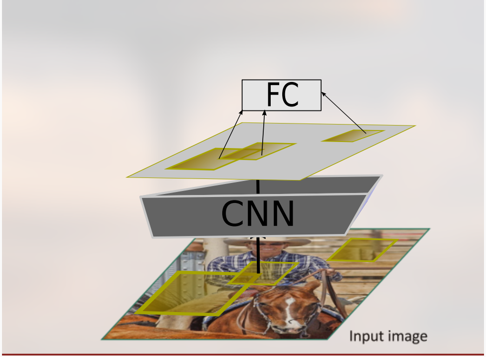
Fast RCNN: Pipeline



Fast RCNN: Pipeline [3].

Project RoI to the feature map [6, Appendix A].

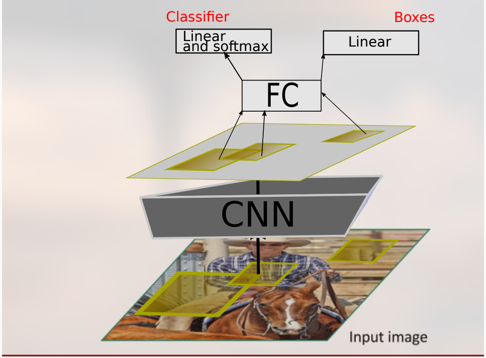
Fast RCNN: Pipeline



Fast RCNN: Pipeline [3].

Warp images with a pooling layer and feed foward a fully connected network.

Fast RCNN: Pipeline



Fast RCNN: Pipeline [3].

Compute, for each RoI, the softmax probabilities and bounding-box regression offsets.

Fast RCNN: Training

The Fast RCNN framework can be trained with the backpropagation algorithm; for this, we need a loss function for a multi-task (classification and regression) method; this is described in the next definition.

Fast RCNN: Training

Loss function
Let $\mathcal{L} = I_N$ be set of of labels, with $0 \in \mathcal{L}$ being the background, and

$$\mathsf{X} = \{(t^{(i)}, k^{(i)}), 1 \leq i \leq n\},$$

in which $t^{(i)} = (t_x^{(i)}, t_y^{(i)}, t_w^{(i)}, t_h^{(i)})^3$ is equivalent to the bounding box of class y_i, be the input data. Let, then, $(t^{(i)}, w^{(i)}) \in \mathsf{X}$ be an instance and write $\hat{t}^{(i)}$ and p for the network output for the bounding box and for the probabilities vector. In this circunstances, the function, for some hyperparameter $\lambda > 0$,

$$L(\hat{\mathrm{p}}, w^{(i)}, \hat{t}^{(i)}, \mathrm{t}^{(\mathrm{i})}) = -\log \mathrm{p}_{w^{(i)}} + \lambda[w^{(i)} \geq 1] \sum_{j \in \{x,y,w,h\}} \mathrm{smooth}_{L_1}(\hat{t}_j^{(i)} - t_j^{(i)})$$

(we write $P \mapsto [P]$ for the Iverson bracket and $\mathrm{smooth}_{L_1}(x) = [|x| < 1] \cdot x^2/2 + [|x| \geq 1] \cdot (|x| - .5)$ is our loss function).

[3] This is a log-scale transforamtion of the inputs; see [4, Appendix C].

Fast RCNN: Summary

The framework of Fast RCNN introduced a single stage fully trainable framework for object detection; the training, in particular, was based on a multi-task loss function. However, it still uses an independent heuristic system for region proposals; next architecture, then, aims to address this scenario.

Faster RCNN

Currently, we are using a convolutional neural networks to both classify the regions and to fix the region proposals; however, we still need a external tool to propose the regions. The next step, then, is to leave

to the network the task of proposing regions!

This is precisely the increment introduced with the Faster RCNN framework: it introduces a *Region Proposal Network* (RPN), which produces a single netowrk that concentrates the full pipeline.

Faster RCNN: Architecture

In Fast RCNN, we fowarded the whole image in the CNN and, then, projected the externally proposed RoIs in the feature map; now, we continue to foward the whole image, but we insert another CNN (possibly VGG [12]) fine-tuned to propose regions.

Faster RCNN: Architecture

This is the (essential component of the) framework:

- you initally assign (to each pixel in the feature map) a set of boxes, called *anchor boxes*;

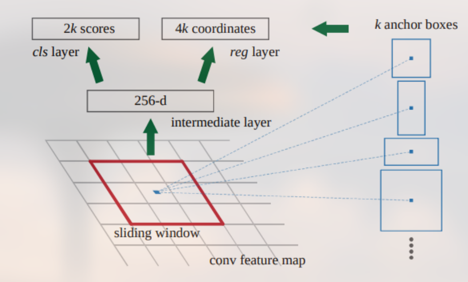


Figure 8: Anchor boxes for Faster RCNN [11].

Faster RCNN: Architecture

This is the (essential component of the) framework:

- you initally assign (to each pixel in the feature map) a set of boxes, called *anchor boxes*;
- in particular, you have to choose its scale (area) and its aspect ratio (ratio between width and height);

Faster RCNN: Architecture

This is the (essential component of the) framework:

- you initally assign (to each pixel in the feature map) a set of boxes, called *anchor boxes*;
- in particular, you have to choose its scale (area) and its aspect ratio (ratio between width and height);
- those boxes are forwarded to a CNN (which involves choosing the size of the initial convolutional layer) that identifies whether the box contains a object and, in this case, tries to refine the bounding box;

Faster RCNN: Architecture

This is the (essential component of the) framework:

- you initally assign (to each pixel in the feature map) a set of boxes, called *anchor boxes*;
- in particular, you have to choose its scale (area) and its aspect ratio (ratio between width and height);
- those boxes are forwarded to a CNN (which involves choosing the size of the initial convolutional layer) that identifies whether the box contains a object and, in this case, tries to refine the bounding box;
- the proposals, then, are used with Fast RCNN to classify objects.

Faster RCNN: Architecture

This is the (essential component of the) framework:

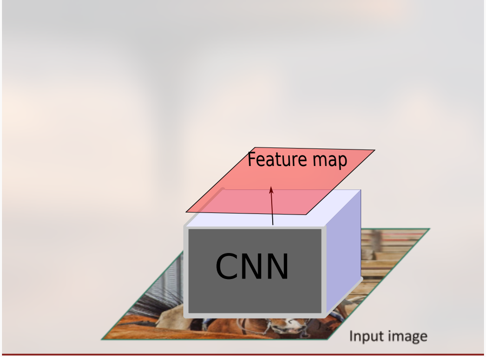
- you initally assign (to each pixel in the feature map) a set of boxes, called *anchor boxes*;
- in particular, you have to choose its scale (area) and its aspect ratio (ratio between width and height);
- those boxes are forwarded to a CNN (which involves choosing the size of the initial convolutional layer) that identifies whether the box contains a object and, in this case, tries to refine the bounding box;
- the proposals, then, are used with Fast RCNN to classify objects.
- Tipically, many bounding boxes have a large intersection area; in those cases, we need to make a choice (usually with non-maximum supression (NMS), which selects the most probable box) between one of them.

Faster RCNN: Architecture

Faster RCNN: Pipeline [11].

Initial image.

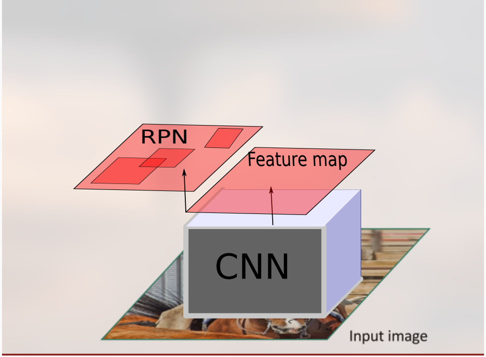
Faster RCNN: Architecture



Faster RCNN: Pipeline [11].

Forward the input image in a CNN.

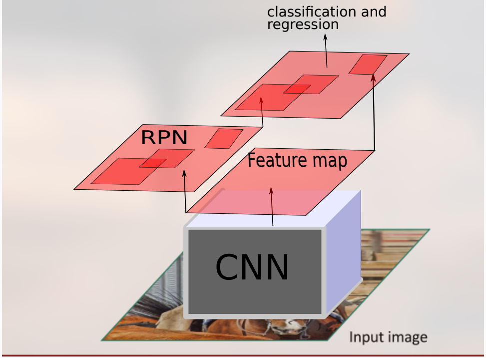
Faster RCNN: Architecture



Faster RCNN: Pipeline [11].

Generate region proposals with the Region Proposal Network (RPN); it makes a binary classification (verifies whether it contains an object) for each anchor box.

Faster RCNN: Architecture



Faster RCNN: Pipeline [11].

Classify the proposed regions accordingly to a predefined set of classes.

Faster RCNN: Training

The training is truly abstruse; explicitly, we have a jointly training of two networks (RPN and Fast RCNN) with four multi-task losses.

- RPN binary classification (`cross entropy`);
- RPN regression (`SmoothL1`);
- Fast RCNN classification (`cross entropy`);
- Fast RCNN regression (`SmoothL1`);

Faster RCNN: Training

The training is truly abstruse; explicitly, we have a jointly training of two networks (RPN and Fast RCNN) with four multi-task losses.

- RPN binary classification (`cross entropy`);
- RPN regression (`SmoothL1`);
- Fast RCNN classification (`cross entropy`);
- Fast RCNN regression (`SmoothL1`);

The authors of [11] of Faster RCNN proposed three distinct ways for this joint training, but, for convenience, we will skip it. ☺

Faster RCNN: Summary

In Faster RCNN, the authors introduced a deep learning based method for making the proposals; with this, they designed a end-to-end system for localization and recognition of objects in images. In this presentation, we will use it and `PyTorch` in a experiment.

Faster RCNN: Summary

In Faster RCNN, the authors introduced a deep learning based method for making the proposals; with this, they designed a end-to-end system for localization and recognition of objects in images. In this presentation, we will use it and `PyTorch` in a experiment.

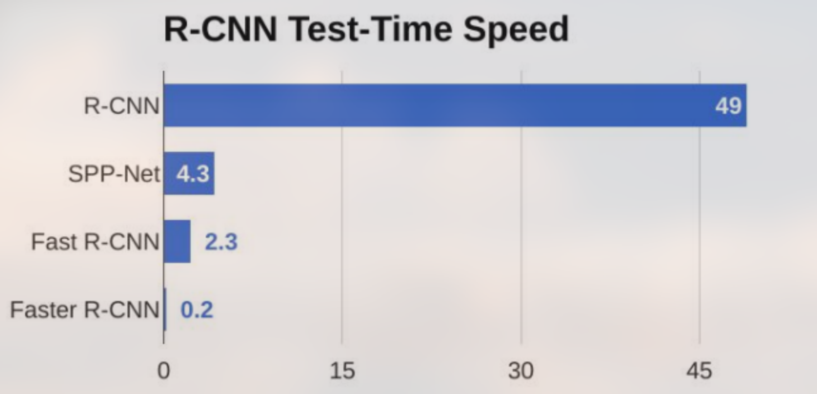


Figure 9: RCNN test-time speed [8].

Automatic Chart Analysis

What?

Automatic interpretation of charts is an (active) research area that aims to circumvent humans from

1. the identification (and possibly the extraction) of charts in documents (both scanned and digitally-born);

What?

Automatic interpretation of charts is an (active) research area that aims to circumvent humans from

1. the identification (and possibly the extraction) of charts in documents (both scanned and digitally-born);
2. the classification of chart images according to a pre-defined set of taxonomies;

What?

Automatic interpretation of charts is an (active) research area that aims to circumvent humans from

1. the identification (and possibly the extraction) of charts in documents (both scanned and digitally-born);
2. the classification of chart images according to a pre-defined set of taxonomies;
3. the segmentation of multi-panel charts for posterior processing;

What?

Automatic interpretation of charts is an (active) research area that aims to circumvent humans from

1. the identification (and possibly the extraction) of charts in documents (both scanned and digitally-born);
2. the classification of chart images according to a pre-defined set of taxonomies;
3. the segmentation of multi-panel charts for posterior processing;
4. the generation of the data used to draw the chart (reverse engineer it!).

Why?

Some applications of this area would truly impact society and, in particular, the academy! By the way, a somewhat exhaustive survey is available in [1].

1. Redesign of charts: the internet (and PowerPoint) estimulated the spread of perceptually inappropriate visualizations; if we could gather their data, we could redesign them.

Figure 10: A doughnut chart with seemingly random numbers.

Why?

Some applications of this area would truly impact society and, in particular, the academy! By the way, a somewhat exhaustive survey is available in [1].

1. Redesign of charts: the internet (and PowerPoint) estimulated the spread of perceptually inappropriate visualizations; if we could gather their data, we could redesign them.
2. Acessibility: currently, acessibility systems use human generated tags to describe images; armed with the data, we could automatically generate a chart description, aiding visually impaired users.

Why?

Some applications of this area would truly impact society and, in particular, the academy! By the way, a somewhat exhaustive survey is available in [1].

1. Redesign of charts: the internet (and PowerPoint) estimulated the spread of perceptually inappropriate visualizations; if we could gather their data, we could redesign them.
2. Acessibility: currently, acessibility systems use human generated tags to describe images; armed with the data, we could automatically generate a chart description, aiding visually impaired users.
3. Chart Retrieval: sometimes, charts contain informations that could be of some use to search engines; reverse engineering visualization, than, could assist the design of those systems;

Why?

Some applications of this area would truly impact society and, in particular, the academy! By the way, a somewhat exhaustive survey is available in [1].

1. Redesign of charts: the internet (and PowerPoint) estimulated the spread of perceptually inappropriate visualizations; if we could gather their data, we could redesign them.
2. Acessibility: currently, acessibility systems use human generated tags to describe images; armed with the data, we could automatically generate a chart description, aiding visually impaired users.
3. Chart Retrieval: sometimes, charts contain informations that could be of some use to search engines; reverse engineering visualization, than, could assist the design of those systems;
4. Bibliometrics: charts have a main role in science; we could, in scale, identifty, within this area, popular chart designs across the literature, aiding potential researchers in their own visualizations.

A motivation

In this presentation, we will address the generation of the data used to draw the chart; specifcally, we will design a protocol that detects legends. Currently, we have a system that generates, using the text data [9], the full specification of the chart; we also have algorithms to, given the legend localization, extract the color mapping used in the plot [10]. The automatic detection of legends, then, would allow us to assemble those frameworks in a end-to-end pipeline for chart specification generation.

Conclusions: what lies ahead?

Conclusions: what lies ahead?

There is still a LOT of scenarios that should be addressed in order to design an appropriate framework for chart mining.

1. Generation of (annotated and public) data sets;

Conclusions: what lies ahead?

There is still a LOT of scenarios that should be addressed in order to design an appropriate framework for chart mining.

1. Generation of (annotated and public) data sets;
2. Generalization of current protocols;

Conclusions: what lies ahead?

There is still a LOT of scenarios that should be addressed in order to design an appropriate framework for chart mining.

1. Generation of (annotated and public) data sets;
2. Generalization of current protocols;
3. Adaptation of current state-of-the-art computer vision tasks to the idiosyncrasies of data visualization;

Conclusions: what lies ahead?

There is still a LOT of scenarios that should be addressed in order to design an appropriate framework for chart mining.

1. Generation of (annotated and public) data sets;
2. Generalization of current protocols;
3. Adaptation of current state-of-the-art computer vision tasks to the idiosyncrasies of data visualization;
4. Implementation of fully trainable methods (for instance, Neural Networks).

Questions?

References i

K. Davila, S. Setlur, D. Doermann, B. U. Kota, and V. Govindaraju.
Chart mining: A survey of methods for automated chart analysis.
43(11):3799–3819, Nov. 2021.

P. F. Felzenszwalb and D. P. Huttenlocher.
Efficient graph-based image segmentation.
59(2):167–181, Sept. 2004.

R. Girshick.
Fast r-cnn, 2015.

R. Girshick, J. Donahue, T. Darrell, and J. Malik.
Rich feature hierarchies for accurate object detection and semantic segmentation, 2014.

References ii

K. He, G. Gkioxari, P. Dollár, and R. Girshick.
Mask r-cnn, 2018.

K. He, X. Zhang, S. Ren, and J. Sun.
Spatial pyramid pooling in deep convolutional networks for visual recognition.
Lecture Notes in Computer Science, page 346–361, 2014.

A. Krizhevsky, I. Sutskever, and G. E. Hinton.
Imagenet classification with deep convolutional neural networks.
In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

F.-F. Li, J. Johnson, and S. Yeung.
Lecture 11: Detection and segmentation, 2017.

References iii

J. Poco and J. Heer.
Reverse-engineering visualizations: Recovering visual encodings from chart images.
36(3):353–363, June 2017.

J. Poco, A. Mayhua, and J. Heer.
Extracting and retargeting color mappings from bitmap images of visualizations.
24(1):637–646, Jan. 2018.

S. Ren, K. He, R. Girshick, and J. Sun.
Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

K. Simonyan and A. Zisserman.
Very deep convolutional networks for large-scale image recognition, 2015.

J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders.
Selective search for object recognition.
104(2):154–171, Apr. 2013.

Thanks!