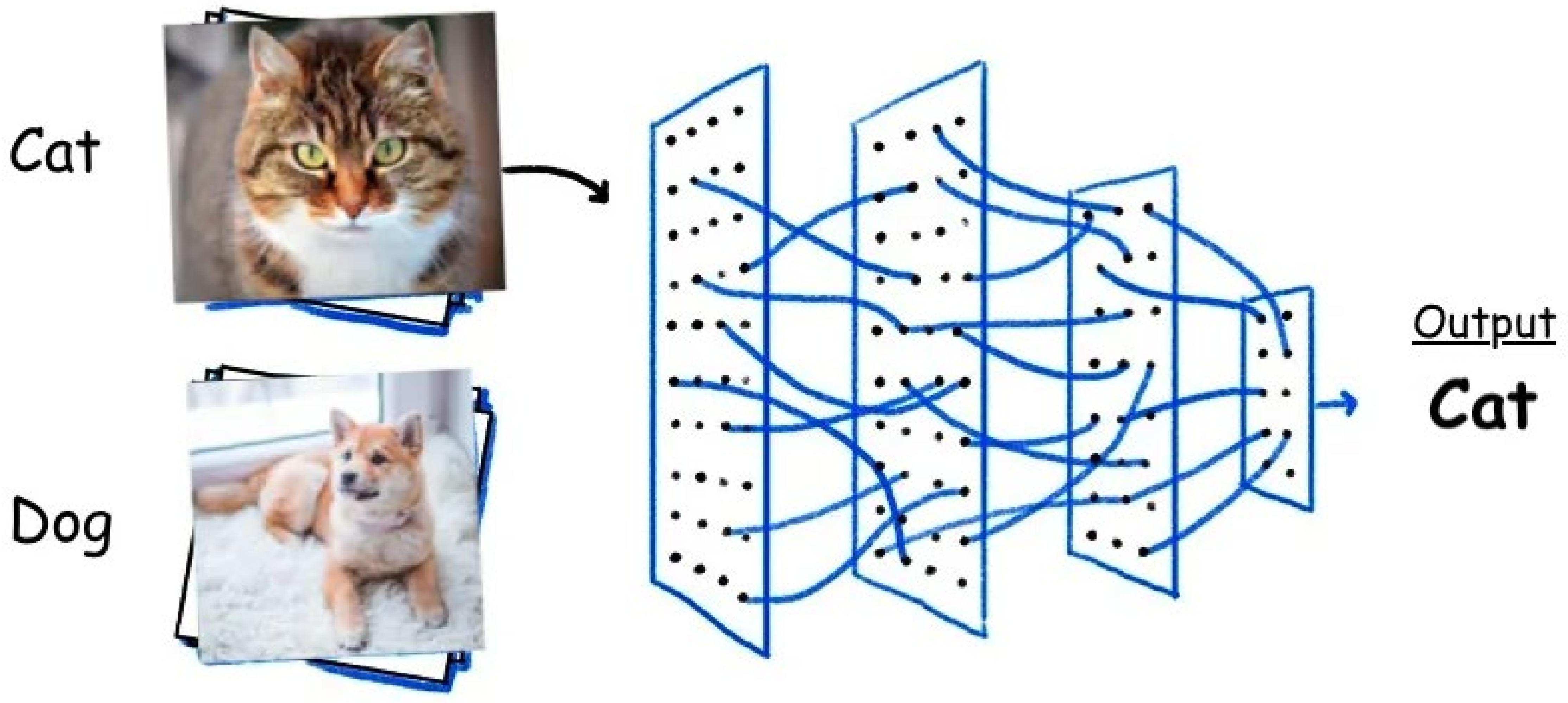
OBJECT DETECTION

– Matee Vadrukchid –

INTRODUCTION

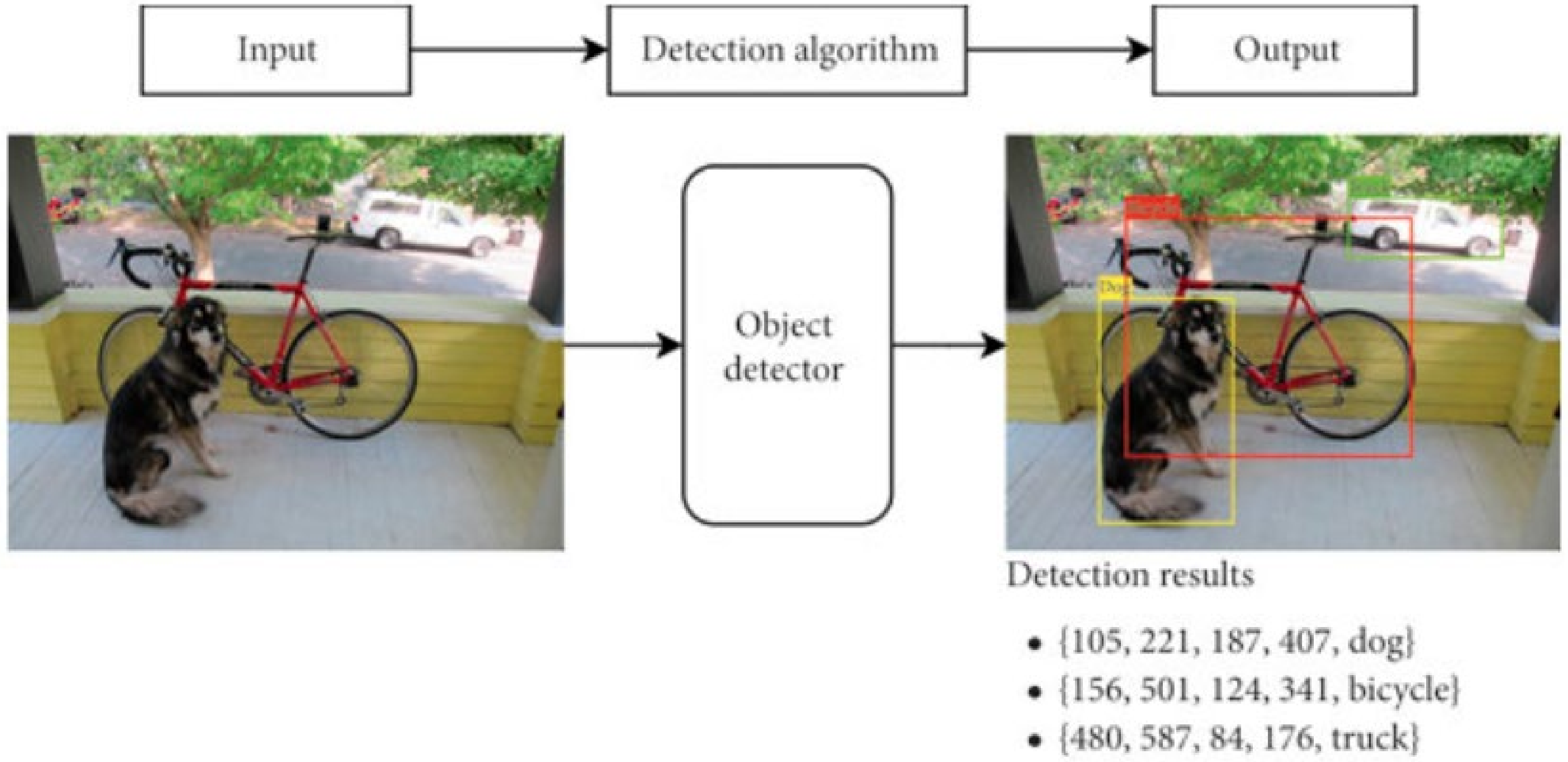
Image Classification Model



Link: https://docs.clarifai.com/tutorials/how -to-evaluate-an-image-classification -model/

INTRODUCTION

Object detection Model



Input

Detection algorithm

Output

Object detector

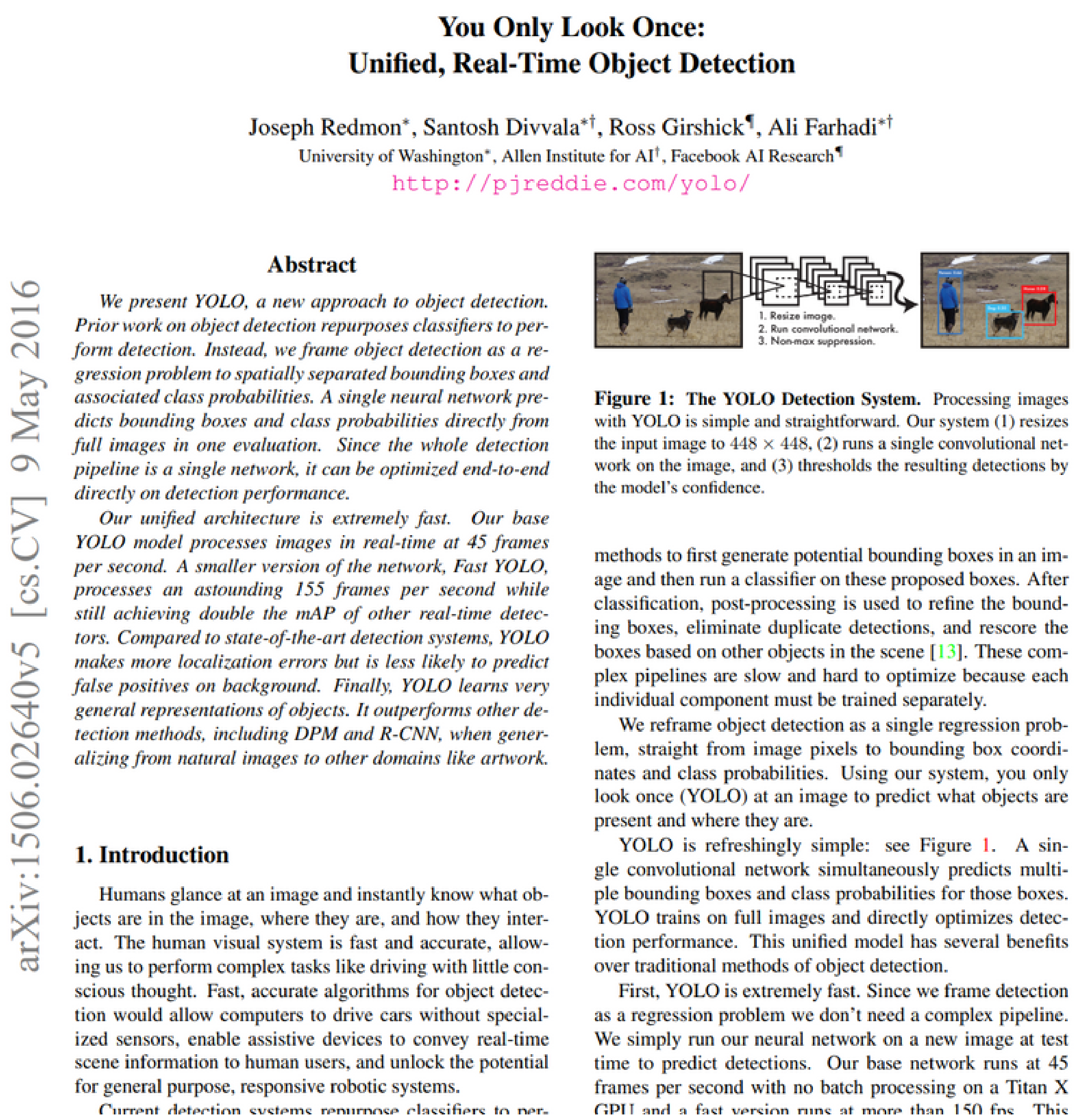
Detection results

- {105, 221, 187, 407, dog}
- {156, 501, 124, 341, bicycle}
- {480, 587, 84, 176, truck}

Link: https://www.researchgate.net/figure/Object-detection-in-an-example-image_fig1_341596604

Part 2: Object Detection

Part 2: Object detection



You Only Look Once: Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶
http://pjreddie.com/yolo/

arXiv:1506.02640v5 [cs.CV] 9 May 2016

Abstract

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

1. Introduction

Humans glance at an image and instantly know what objects are in the image, where they are, and how they interact. The human visual system is fast and accurate, allowing us to perform complex tasks like driving with little conscious thought. Fast, accurate algorithms for object detection would allow computers to drive cars without specialized sensors, enable assistive devices to convey real-time scene information to human users, and unlock the potential for general purpose, responsive robotic systems.

Current detection systems repurpose classifiers to per-

Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

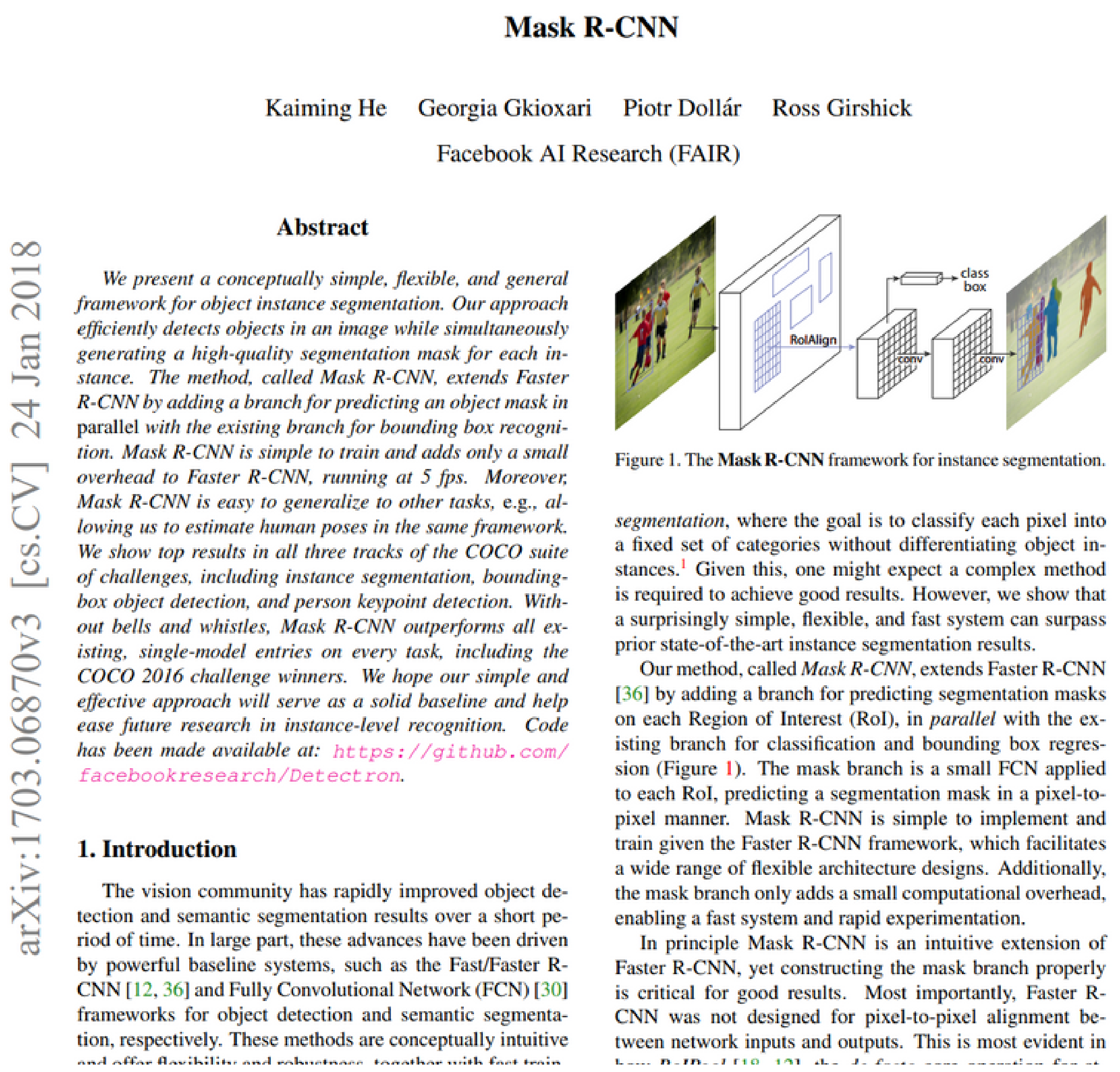
methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene [13]. These complex pipelines are slow and hard to optimize because each individual component must be trained separately.

We reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Using our system, you only look once (YOLO) at an image to predict what objects are present and where they are.

YOLO is refreshingly simple: see Figure 1. A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection.

First, YOLO is extremely fast. Since we frame detection as a regression problem we don't need a complex pipeline. We simply run our neural network on a new image at test time to predict detections. Our base network runs at 45 frames per second with no batch processing on a Titan X GPU and a fast version runs at more than 150 fps. This

Yolov1



Mask R-CNN

Kaiming He Georgia Gkioxari Piotr Dollár Ross Girshick
Facebook AI Research (FAIR)

arXiv:1703.06870v3 [cs.CV] 24 Jan 2018

Abstract

We present a conceptually simple, flexible, and general framework for object instance segmentation. Our approach efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance. The method, called Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in parallel *with the existing branch for bounding box recognition. Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps. Moreover, Mask R-CNN is easy to generalize to other tasks, e.g., allowing us to estimate human poses in the same framework. We show top results in all three tracks of the COCO suite of challenges, including instance segmentation, bounding-box object detection, and person keypoint detection. Without bells and whistles, Mask R-CNN outperforms all existing, single-model entries on every task, including the COCO 2016 challenge winners. We hope our simple and effective approach will serve as a solid baseline and help ease future research in instance-level recognition. Code has been made available at:* https://github.com/facebookresearch/Detectron.

1. Introduction

The vision community has rapidly improved object detection and semantic segmentation results over a short period of time. In large part, these advances have been driven by powerful baseline systems, such as the Fast/Faster R-CNN [12, 36] and Fully Convolutional Network (FCN) [30] frameworks for object detection and semantic segmentation, respectively. These methods are conceptually intuitive and offer flexibility and robustness, together with fast train-

Figure 1. The **Mask R-CNN** framework for instance segmentation.

segmentation, where the goal is to classify each pixel into a fixed set of categories without differentiating object instances.[1] Given this, one might expect a complex method is required to achieve good results. However, we show that a surprisingly simple, flexible, and fast system can surpass prior state-of-the-art instance segmentation results.

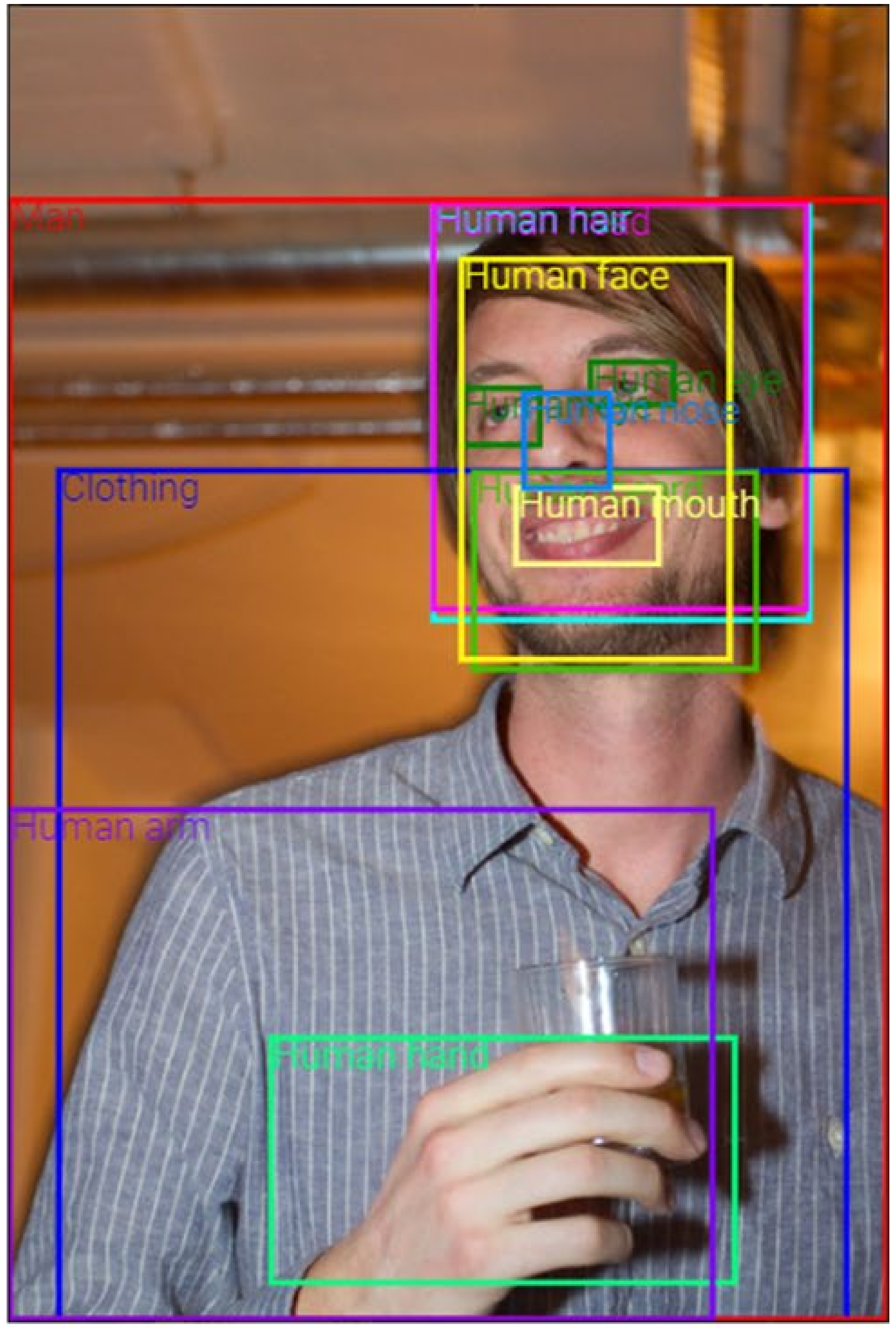
Our method, called *Mask R-CNN*, extends Faster R-CNN [36] by adding a branch for predicting segmentation masks on each Region of Interest (RoI), in *parallel* with the existing branch for classification and bounding box regression (Figure 1). The mask branch is a small FCN applied to each RoI, predicting a segmentation mask in a pixel-to-pixel manner. Mask R-CNN is simple to implement and train given the Faster R-CNN framework, which facilitates a wide range of flexible architecture designs. Additionally, the mask branch only adds a small computational overhead, enabling a fast system and rapid experimentation.

In principle Mask R-CNN is an intuitive extension of Faster R-CNN, yet constructing the mask branch properly is critical for good results. Most importantly, Faster R-CNN was not designed for pixel-to-pixel alignment between network inputs and outputs. This is most evident in how *RoIPool* [18, 12], the *de facto* core operation for at-

Mask-RCNN

Part 2: Object detection

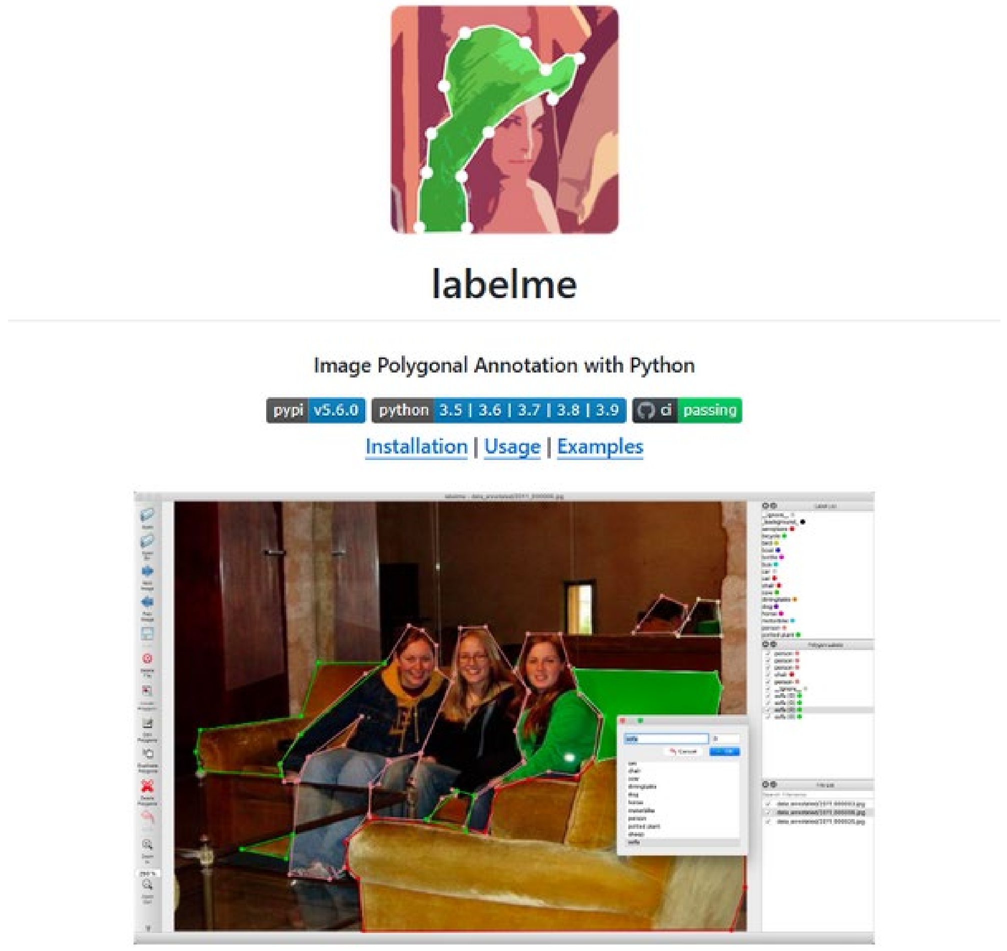
Drawing by Hand



Part 2: Object detection

Labelme

Link: https://github.com/wkentaro/labelme



labelme

Image Polygonal Annotation with Python

pypi v5.6.0 | python 3.5 | 3.6 | 3.7 | 3.8 | 3.9 | ci passing

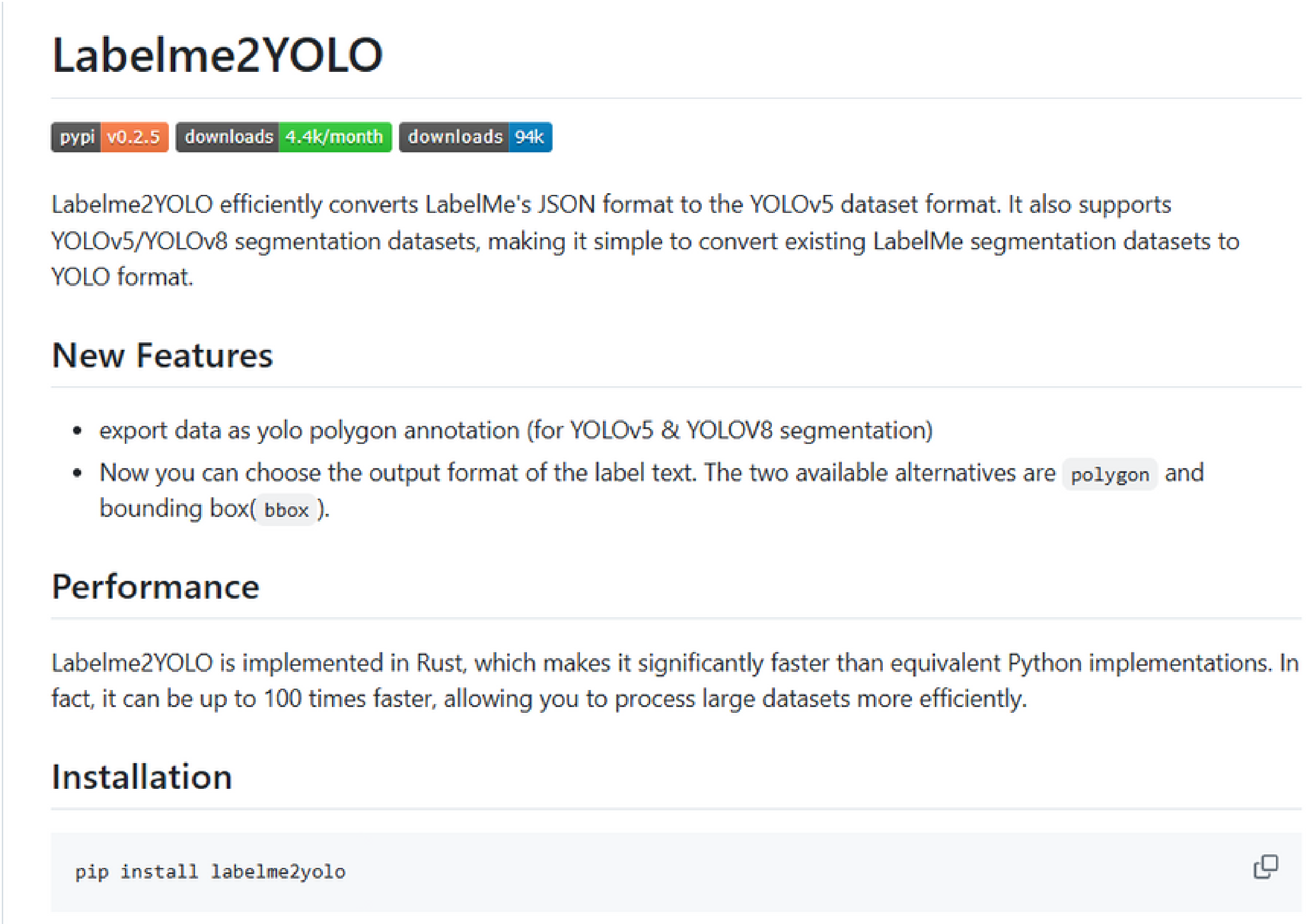
Installation | Usage | Examples

```
pip install labelme
```

Part 2: Object detection

Labelme2YOLO

Link: https://github.com/GreatV/labelme2yolo



Labelme2YOLO

Labelme2YOLO efficiently converts LabelMe's JSON format to the YOLOv5 dataset format. It also supports YOLOv5/YOLOv8 segmentation datasets, making it simple to convert existing LabelMe segmentation datasets to YOLO format.

New Features

- export data as yolo polygon annotation (for YOLOv5 & YOLOV8 segmentation)
- Now you can choose the output format of the label text. The two available alternatives are `polygon` and bounding box(`bbox`).

Performance

Labelme2YOLO is implemented in Rust, which makes it significantly faster than equivalent Python implementations. In fact, it can be up to 100 times faster, allowing you to process large datasets more efficiently.

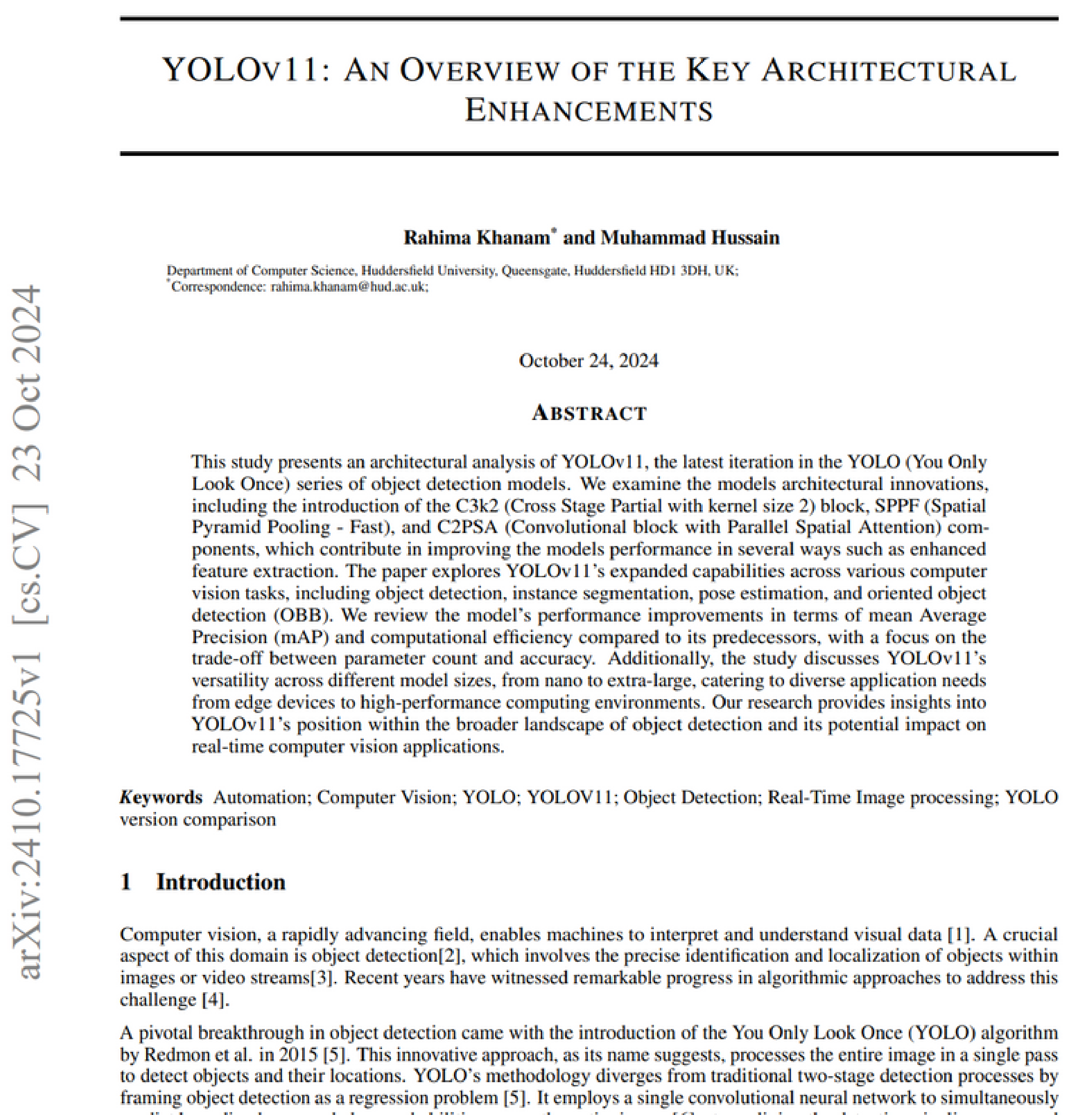
Installation

```
pip install labelme2yolo
```

Part 2: Object detection

YOLOV11

Install : pip install ultralytics



arXiv:2410.17725v1 [cs.CV] 23 Oct 2024

YOLOv11: An Overview of the Key Architectural Enhancements

Rahima Khanam* **and Muhammad Hussain**

Department of Computer Science, Huddersfield University, Queensgate, Huddersfield HD1 3DH, UK;
*Correspondence: rahima.khanam@hud.ac.uk;

October 24, 2024

Abstract

This study presents an architectural analysis of YOLOv11, the latest iteration in the YOLO (You Only Look Once) series of object detection models. We examine the models architectural innovations, including the introduction of the C3k2 (Cross Stage Partial with kernel size 2) block, SPPF (Spatial Pyramid Pooling - Fast), and C2PSA (Convolutional block with Parallel Spatial Attention) components, which contribute in improving the models performance in several ways such as enhanced feature extraction. The paper explores YOLOv11's expanded capabilities across various computer vision tasks, including object detection, instance segmentation, pose estimation, and oriented object detection (OBB). We review the model's performance improvements in terms of mean Average Precision (mAP) and computational efficiency compared to its predecessors, with a focus on the trade-off between parameter count and accuracy. Additionally, the study discusses YOLOv11's versatility across different model sizes, from nano to extra-large, catering to diverse application needs from edge devices to high-performance computing environments. Our research provides insights into YOLOv11's position within the broader landscape of object detection and its potential impact on real-time computer vision applications.

Keywords Automation; Computer Vision; YOLO; YOLOV11; Object Detection; Real-Time Image processing; YOLO version comparison

1 Introduction

Computer vision, a rapidly advancing field, enables machines to interpret and understand visual data [1]. A crucial aspect of this domain is object detection[2], which involves the precise identification and localization of objects within images or video streams[3]. Recent years have witnessed remarkable progress in algorithmic approaches to address this challenge [4].

A pivotal breakthrough in object detection came with the introduction of the You Only Look Once (YOLO) algorithm by Redmon et al. in 2015 [5]. This innovative approach, as its name suggests, processes the entire image in a single pass to detect objects and their locations. YOLO's methodology diverges from traditional two-stage detection processes by framing object detection as a regression problem [5]. It employs a single convolutional neural network to simultaneously

Link :
https://arxiv.org/pdf/2410.17725

Link :https://docs.ultralytics.com/modes/train/#train -settings

Link:
https://github.com/ultralytics/ultralytics