

# AI/ML Research

## Introduction

Computer Vision (CV) is defined by IBM as the utilization of “machine learning and neural networks to teach computers and systems to derive meaningful information from digital images, videos, and other inputs”. Some notes about CV, it requires a large amount of data which it utilizes to ascertain the “distinctions and ultimately recognize images.”

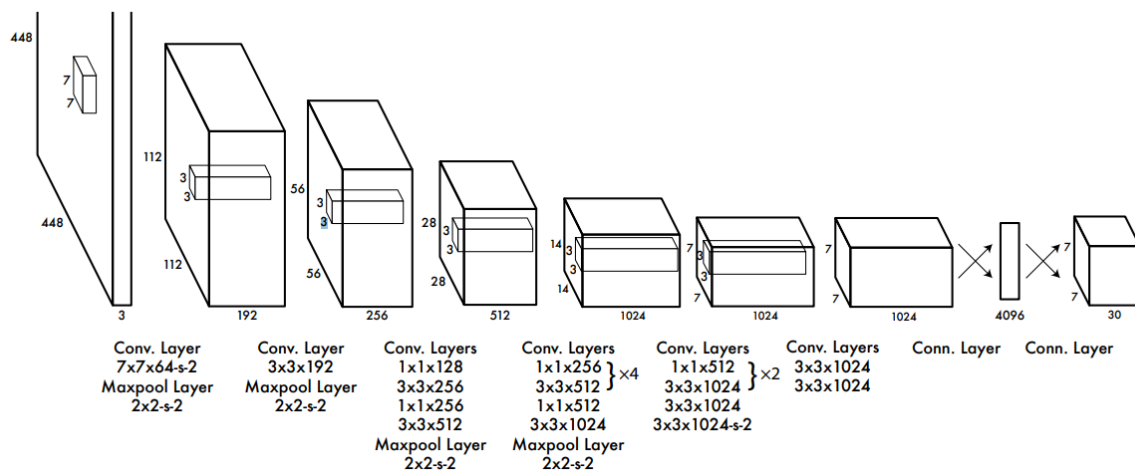
The current main approach for CV is dominated by deep learning, specifically convolutional neural networks (CNN). Generally, Object Detection can be broken into “two categories based on how many times the same input image is passed through the network”. There is Two-Stage/Proposal which consists of some of the likes of: R-CNN, Fast R-CNN, Faster R-CNN, RFCN, and Mask R-CNN. One-Stage/Proposal-Free which consists of YOLO and SSD.

<https://www.v7labs.com/blog/yolo-object-detection>

---

## YOLO

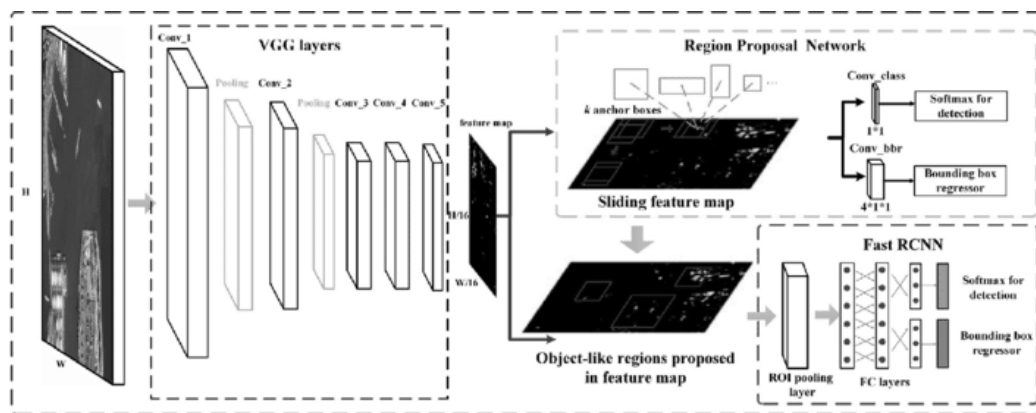
YOLO (You Only Look Once), an algorithm which divides the input image into a grid of cells, and predicts the probability of the presence of an object and bounding box of the object.



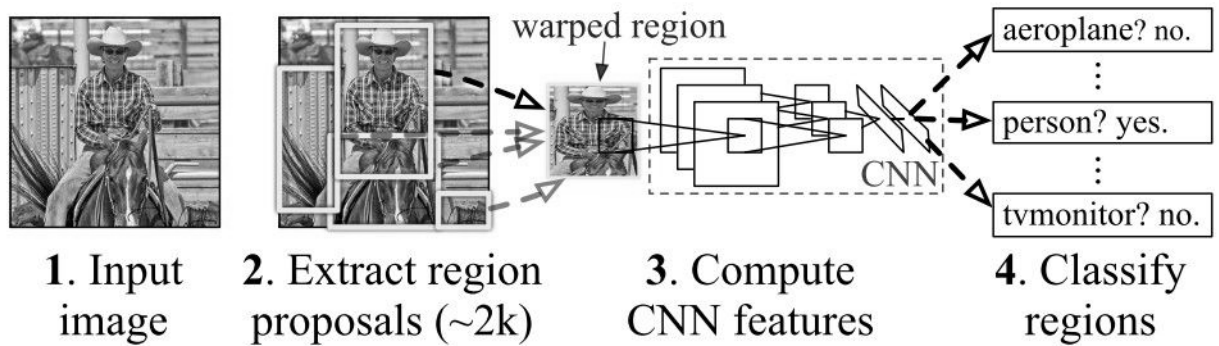
<https://arxiv.org/pdf/1506.02640> [YOLO]

## Faster R-CNN

Faster R-CNN (Region-based Convolutional Neural Network) works by utilizing the Region Proposal Network (RPN) which scans the image through a fixed length features vector which extracts each region. These are then passed through a CNN to perform identification of objects.



[https://www.researchgate.net/figure/The-architecture-of-Faster-R-CNN\\_fig2\\_324903264](https://www.researchgate.net/figure/The-architecture-of-Faster-R-CNN_fig2_324903264)



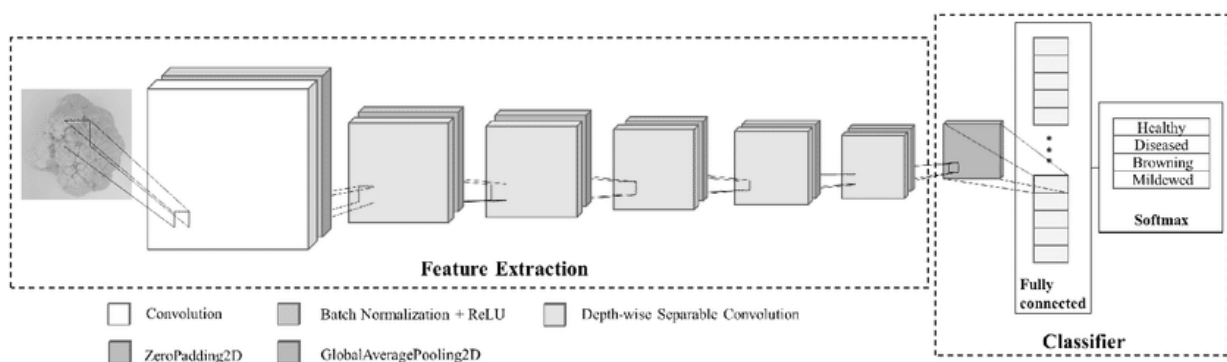
<https://www.digitalocean.com/community/tutorials/faster-r-cnn-explained-object-detection>

<https://arxiv.org/pdf/1506.01497> [Faster RCNN]

<https://www.ibm.com/topics/computer-vision> [IBM]

## MobileNet

According to Andrew G. Howard and Menglong Zhu, MobileNet is a family of mobile-first computer vision models. MobileNets are described as “small, low-latency, lower power models”



[https://www.researchgate.net/figure/Architecture-of-MobileNet\\_fig3\\_363675522](https://www.researchgate.net/figure/Architecture-of-MobileNet_fig3_363675522)

<https://arxiv.org/pdf/1801.04381> [MobileNet v2]

<https://research.google/blog/mobilenets-open-source-models-for-efficient-on-device-vision/> [MobileNet]

<https://arxiv.org/pdf/2407.01435> [MobileNet SSD]

<https://www.youtube.com/watch?v=UZDiGooFs54> [The moment we stopped understanding AI [AlexNet]]

[https://www.youtube.com/watch?v=n9\\_XyCGr-MI](https://www.youtube.com/watch?v=n9_XyCGr-MI) [YOLOv1 from Scratch]

Very interesting video explaining YOLO in more depth.







YOLOv4-tiny is proposed based on YOLOv4 to simplify the network structure and reduce parameters, which makes it suitable for developing on the mobile and embedded devices. It firstly uses two ResBlock-D modules in the ResNet-D network instead of two CSPBlock modules in YOLOv4-tiny, which reduces the computation complexity.

<https://arxiv.org/pdf/2011.04244> [YOLO-v4-tiny]

[https://www.coderun.ca/programming/2021-10-16\\_darknet\\_fps/](https://www.coderun.ca/programming/2021-10-16_darknet_fps/) [YOLO]

TLDR Summary

How many FPS can you expect to process when working with Darknet, YOLO, and OpenCV?

Device	FPS	Method
 Beaglebone Green	0.005 FPS	Darknet (CPU)
 Raspberry Pi 4	2.4 FPS	OpenCV DNN (CPU)
 Jetson Nano	16.9 FPS	OpenCV DNN + CUDA
 Jetson Xavier NX	46.4 FPS	OpenCV DNN + CUDA
 Jetson AGX	71.5 FPS	Darknet + CUDA
 NVIDIA RTX 2070	209.7 FPS	OpenCV DNN + CUDA

Highlighting from these results the speed of the Raspberry Pi 4 is 2.4 FPS or 416.67 ms to perform an inference on a single frame, this does not include other aspects necessary regarding pre and post processing. But, it does not allow for a sufficient realtime speed analysis.





Result of running yolo-v4 tiny sample model only trained on COCO 2017 dataset.

## Frameworks

The first framework, Darknet, was created by one of the authors of the original YOLO algorithm. According to the github created to host the software, Darknet is described as “an open source neural network framework written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation.”

<https://github.com/pjreddie/darknet>

Another framework is Tensorflow, Tensorflow is described as “an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications.”

<https://github.com/tensorflow/tensorflow>

The last framework of significance is PyTorch, it is used for Tensor computation (like NumPy) with strong GPU acceleration, and Deep neural networks built on a tape-based autograd system. Making it a good fit for CV projects.

<https://github.com/pytorch/pytorch>

Our initial efforts have proved fruitful from utilizing Darknet to train the models for the YOLO-v4 tiny architecture. This paired with conversion and quantization of the model to a tensorflow backend called “tflite”, has enabled us to gather higher performance while not significantly impacting the accuracy.

