Image recognition in java using CNN.

Made by:

Jayed Naoufal.

Ziani Zaynab.

Raihan Najib.

Mentored by:

Mr Bahri Abdelkhalak.

Summary

- Introduction
- CNN
- ANN
- YOLO
- Code Explanation
- Demonstration
- Bibliography

Introduction

Object detection and classification is a computer technology related to computer vision and image processing.

It is used in Computer Vision tasks such as: activity recognition, face detection/recognition, tracking moving objects…

Methods for object detection fall in deep-learning-based approaches.

What are Convolutional Neural Networks(CNNs)?

CNN is a class of deep neural network that is mostly used to do image recognition/classification, object detection… etc.

Steps:

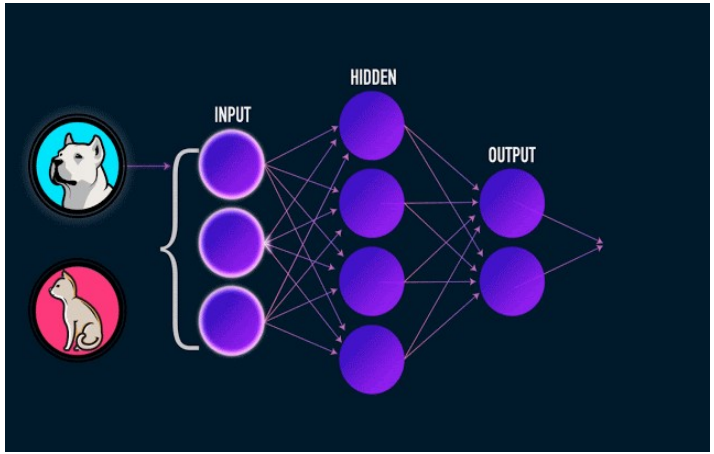
-Convolution

-ReLU (Rectified Linear-Unit)

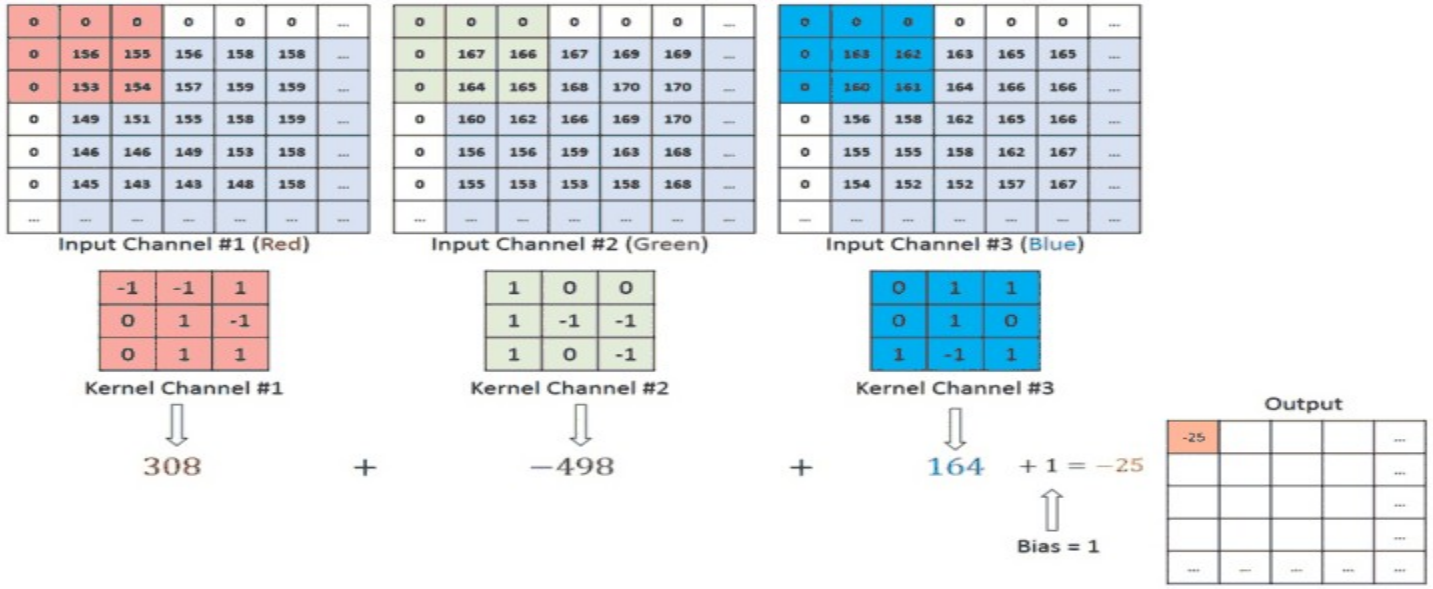
-Pooling

-Flattening

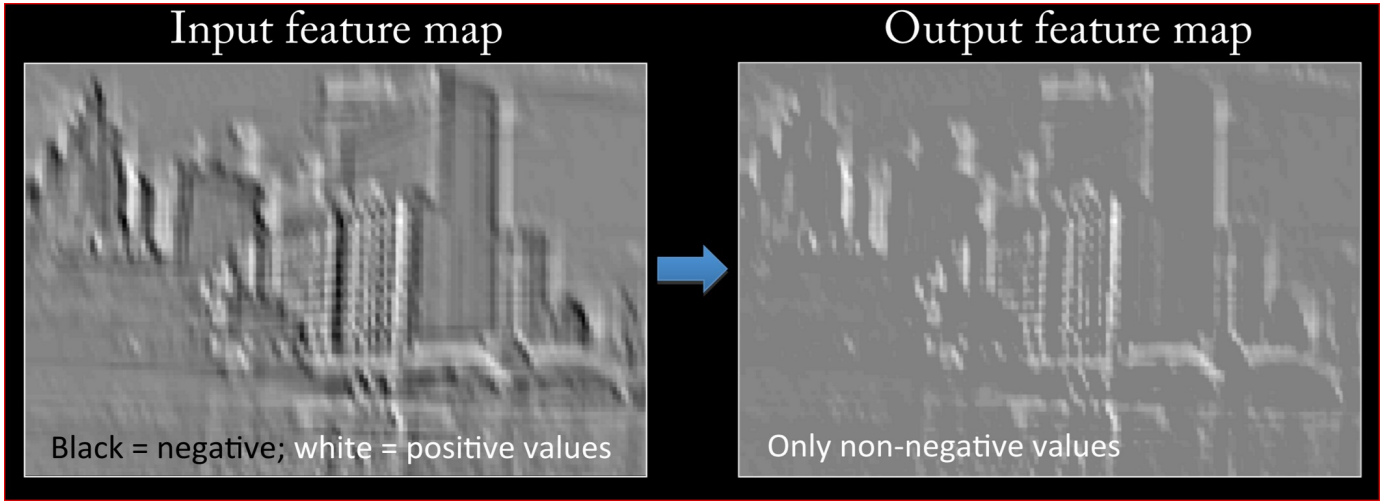
-Full Connection



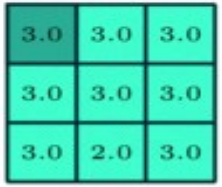
Step1: Convolutions

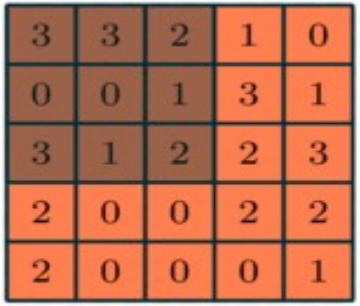


Step2: ReLU (Rectified Linear-Unit)

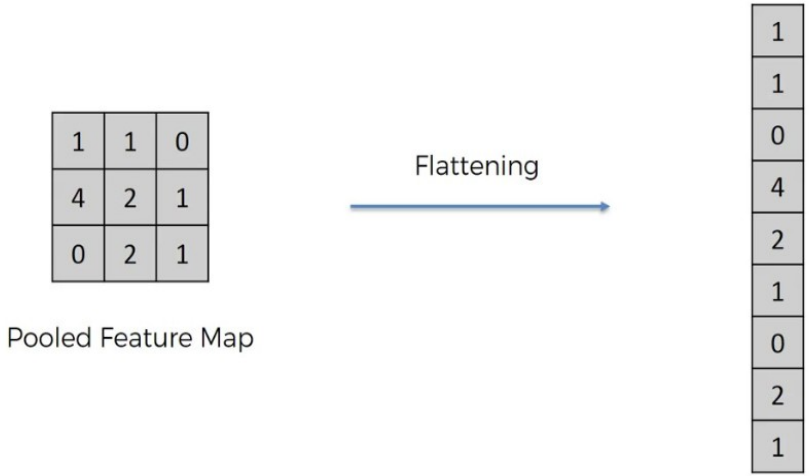


Step3: Pooling

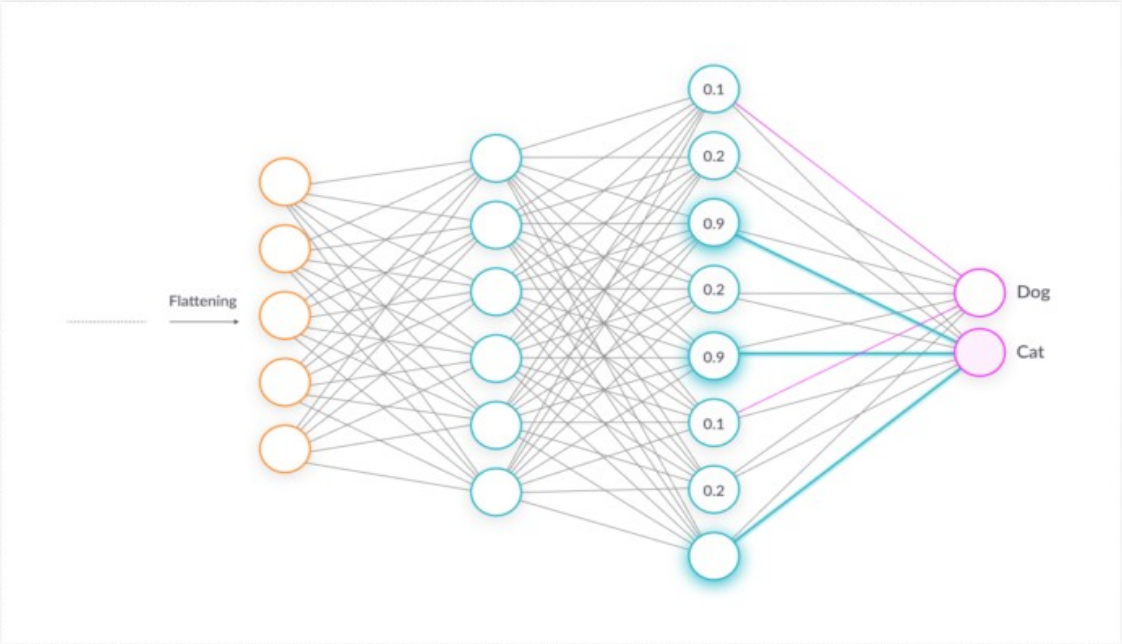




Step4: Flattening



Step5: Full Connection



What are Artificial Neural Networks(ANNs)?

ANN is an information processing paradigm that is inspired by the way the biological nervous system such as the brain processes information. It's composed of a large number of highly interconnected processing elements (neurons) working in union to solve a specific problem.

Activation functions

-Introduces non-linearity to a NN.

- Decides whether a neuron should be activated or not.

-Many activation functions exist, each one has its advantages and drawbacks, we mention: Threshold function, sigmoid function, hyperbolic function (tanh), ReLU.

How do NN learn?

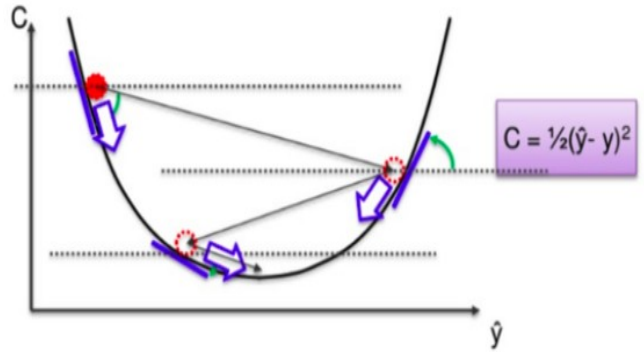
-Neural Networks use the back propagation concept for training.

The Output value is compared to the Actual value, by calculating a loss function : $C = \frac{1}{2}(Y_{output\_value} - Y_{actual\_value})^2$.

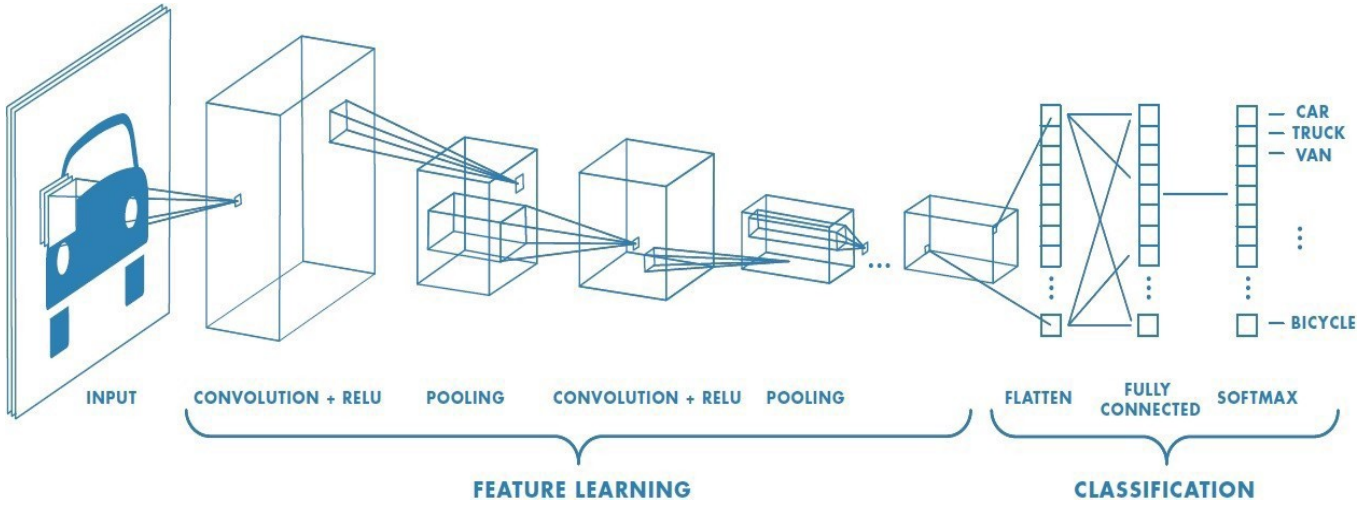
Then information are fed back to the network and weights are being update in the aim of minimizing the loss function.

Gradient descent

-optimization algorithm responsible for finding the minimum cost value, instead of going through each weight at a time we consider the slope in the loss function.



General architecture of CNN

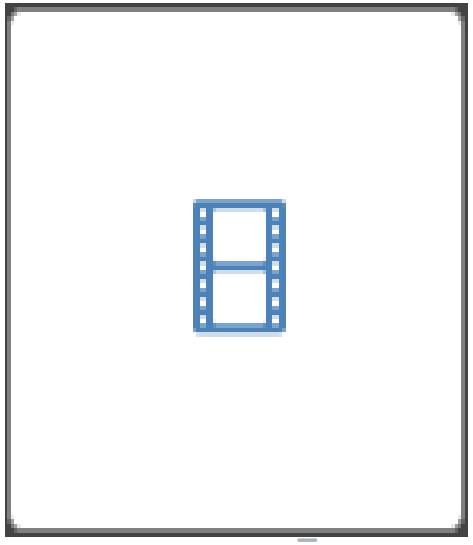


What did we use to recognize images in our project?

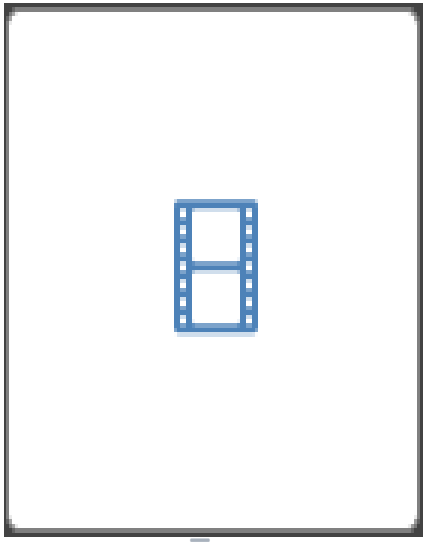
Yolo (You Only Look Once)

You Only Look Once is an algorithm that utilizes a single convolutional network for object detection.

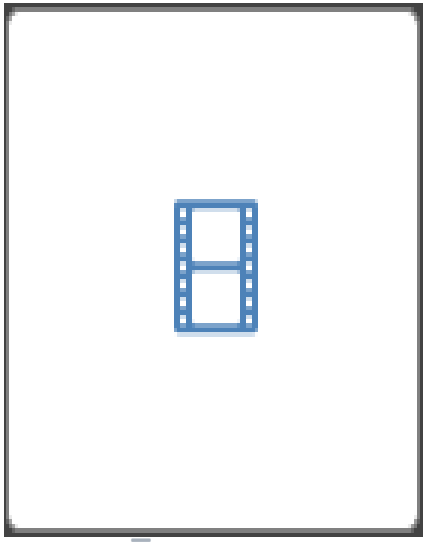
Why exactly YOLO?



Fast RCCNs

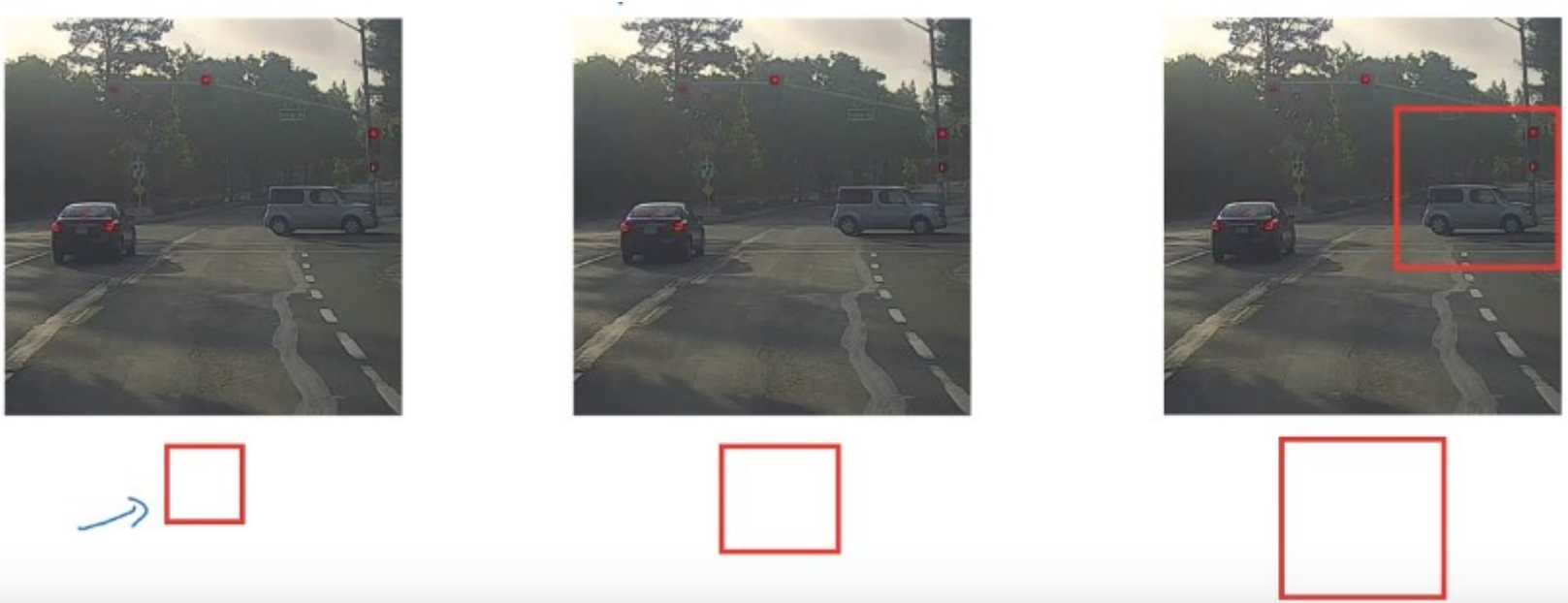


Faster RCCNs



YOLO

Why are fast RCNNs Slower?

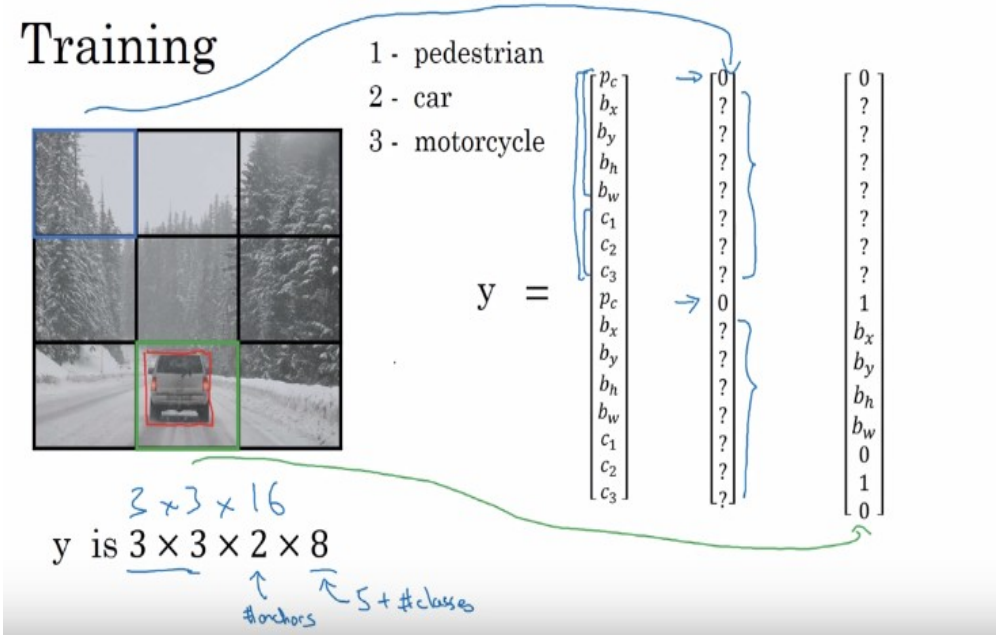


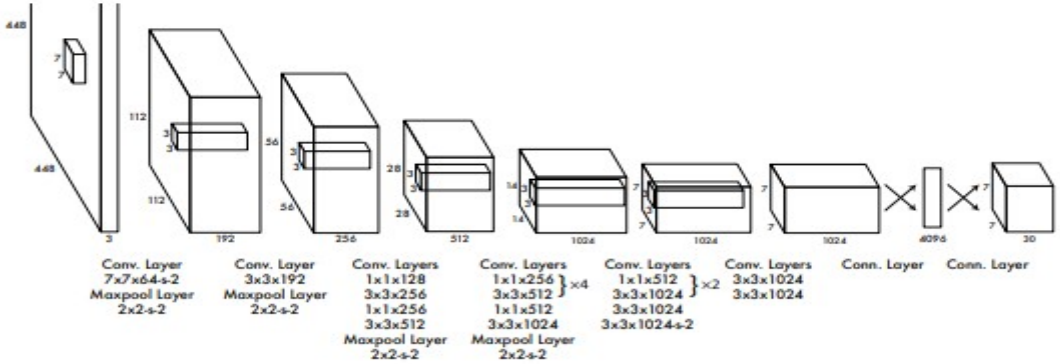
How does YOLO work? What makes it faster?

Training

1 - pedestrian
2 - car
3 - motorcycle

y is $3 \times 3 \times 2 \times 8$





Input

=>

Generating Boxes

=>

What happens after generating the boxes and getting the output?

Final Steps

Removing boxes with a confidence<0.6

Non-Max Suppression

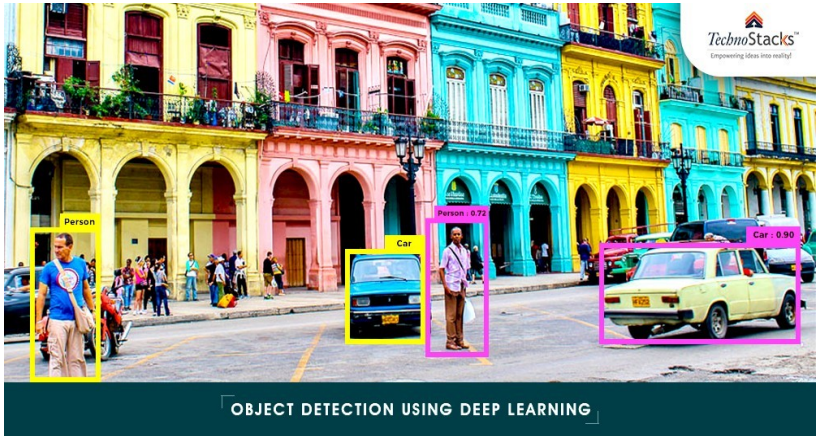
Each output prediction is: p_c, b_x, b_y, b_h, b_w

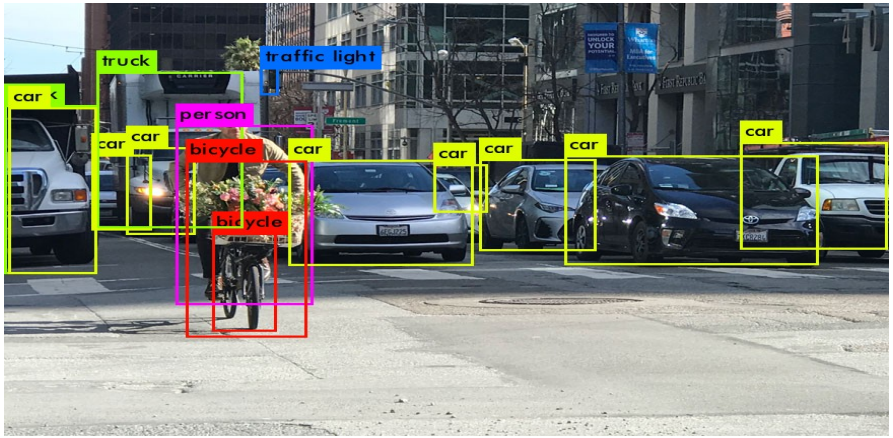
Discard all boxes with $p_c \leq 0.6$

While there are any remaining boxes:

- Pick the box with the largest p_c Output that as a prediction.
- Discard any remaining box with IoU ≥ 0.5 with the box output in the previous step

Some final results





What do we need to run this algorithm?

- OpenCV v3.0 or newer version.
- YOLO v3 object detector model with 3 necessary files:
  - YOLOv3.weights
  - YOLOv3.cfg
  - ObjectsNames.names

Code Explanation

- Loading weights and configuration files.

```
String modelWeights = "C:/yolo/yolov3.weights";      // weights of objects
String modelConfiguration = "C:/yolo/yolov3.cfg";    //Convolutional layers

Net net = Dnn.readNetFromDarknet(modelConfiguration, modelWeights);
Mat image = Imgcodecs.imread(imagepath);
Size sz = new Size( width: 416,  height: 416);
```

- Reading ObjectsNames.names file

```
////////////////////////////////////

String modelNames = "C:/yolo/coco.names";            //Objects names
ArrayList<String> classes = new ArrayList<>();

try {
    FileReader file1 = new FileReader(modelNames);
    BufferedReader bufferedReader = new BufferedReader(file1);
    String Line;
    while ((Line = bufferedReader.readLine()) != null) {
        classes.add(Line);
    }
    bufferedReader.close();
} catch (Exception error) {
    System.out.println(error);
}
```

- Creating a method for extracting the image's layers and their names

Test.java | darknet.java

```
package com.company;

import org.opencv.dnn.Net;
import java.util.ArrayList;
import java.util.List;

public class darknet  {

    static List<String> names;
    static List<Integer> outLayers;
    static List<String> layersNames;

    public darknet(Net net){
        names = new ArrayList<>();
        outLayers = net.getUnconnectedOutLayers().toList();
        layersNames = net.getLayerNames();
    }

    public static List<String> getOutputNames() {
        outLayers.forEach((item) -> names.add(layersNames.get(item - 1)));
        return names;
    }

}
```

- Constructing the Blobs

```
Mat blob = Dnn.blobFromImage(image, scalefactor: 0.00392, sz, new Scalar( v0: 0), swapRB: true, crop: false);
net.setInput(blob);

//////////////////////////
darknet Layers = new darknet(net);
List<String> outBlobNames = darknet.getOutputNames();
//outBlobNames.forEach(System.out::println);
        ////////////////////////////////////////

            // backforward in Boxes

        List<Mat> result = new ArrayList<>();
        net.forward(result, outBlobNames);
        //result.forEach(System.out::println);

        ////////////////////////////////////////
```

- A blob is just a (potentially collection) of image(s) with the same spatial dimensions (i.e., width and height.

- Removing boxes with confidence<0.95f

```
float confThreshold = 0.95f;    ///// Minimum probability to filter weak detections
List<Integer> classesIds = new ArrayList<>();
List<Float> confidences = new ArrayList<>();
List<Rect> rects = new ArrayList<>();

for (int i = 0; i < result.size(); i++) {
    // [center_x, center_y, width, height]+ probabilities
    Mat level = result.get(i);                    ////// generate matrix of 4 dimensions
    for (int j = 0; j < level.rows(); j++) {
            Mat row = level.row(j);
            Mat scores = row.colRange(5, level.cols());
                Core.MinMaxLocResult mm = Core.minMaxLoc(scores);   /// search the row of the maximmum
                    float confidence = (float) mm.maxVal;
                    Point classIdPoint = mm.maxLoc;          /// get location for max condidance

                    if (confidence > confThreshold)     //// filter row of weak detections
        {
            int centerX = (int) (row.get( row: 0,  col: 0)[0] * image.cols());    ///// get
            int centerY = (int) (row.get( row: 0,  col: 1)[0] * image.rows());
            int width = (int) (row.get( row: 0,  col: 2)[0] * image.cols());
            int height = (int) (row.get( row: 0,  col: 3)[0] * image.rows());
            int left = centerX - width / 2;                  //// get topleft point coordinates
            int top = centerY - height / 2;

            classesIds.add((int) classIdPoint.x);
            confidences.add((float) confidence);
            rects.add(new Rect(left, top, width, height));      //// informations of boxes
```

- Non Maximum suppression

```
////////////////////// Non-Maximum-Suppression
Rect[] boxesArray = rects.toArray(new Rect[0]);
MatOfRect boxes = new MatOfRect(boxesArray);
MatOfFloat confidence = new MatOfFloat(Converters.vector_float_to_Mat(confidences));
float NMSThreshold = 0.4f;   ///// This is our non-maxima suppression threshold
MatOfInt indices = new MatOfInt();
Dnn.NMSBoxes(boxes, confidence, confThreshold, NMSThreshold, indices);    /// intersection over union IOU
```

- Draw the final boxes

```
int[] ind = indices.toArray();
System.out.println(ind.length);
for (int i = 0; i < ind.length; i++) {

    System.out.println(ind[i]);
    int idx = ind[i];
    Rect box = boxesArray[idx];     /// get objects by index of classes
    System.out.println(box);

    String label = classes.get(classesIds.get(i)) + String.format("%.2f", confidences.get(i) * 100) + "%";
    Imgproc.rectangle(image, box.tl(), box.br(), new Scalar(0, 255, 0),  thickness: 5);   //// object box
    Imgproc.putText(image, label, new Point( x: box.x + 10, box.y),  fontFace: 5,  fontScale: 5, new Scalar(0, 255, 0),  thickness: 3); // object r
    //System.out.println(box);
}
```

Demonstration

Yolo Image detection

Now we will show you the demonstration of the previously explained java code.

Bibliography

https://en.wikipedia.org/wiki/Object_detection

https://medium.com/nybles/a-brief-guide-to-convolutional-neural-network-cnn-642f47e88ed4

https://medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8

https://towardsdatascience.com/introduction-to-artificial-neural-networks-ann-1aea15775ef9