8/6/2022

Muhammad Osama

U41261655

CS 767 Project Report

Implementation of YOLOV3 for Object Detection

Contents

[Introduction 4](#_Toc110689878)

[Implementation of the Model 6](#_Toc110689879)

[Architecture 6](#_Toc110689880)

[Loss Function 6](#_Toc110689881)

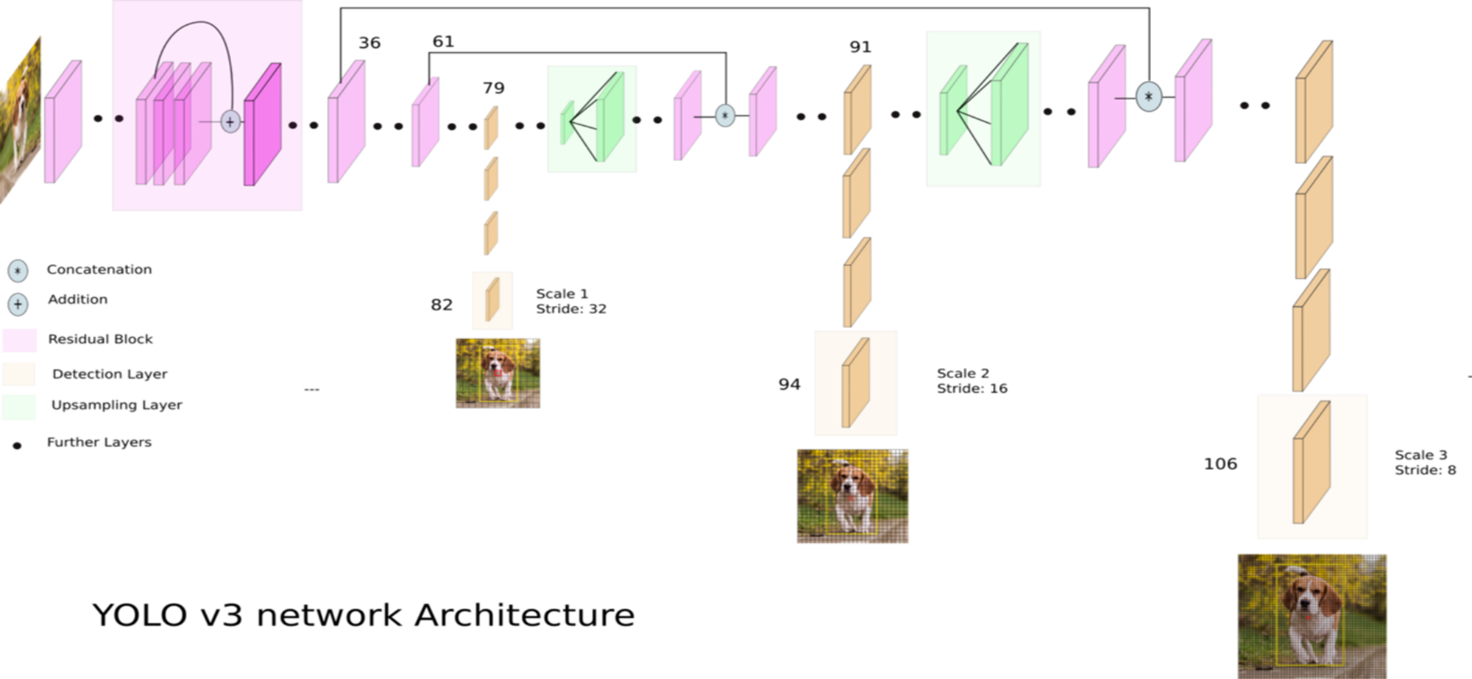
[Dataset Construction 6](#_Toc110689882)

[Training & Testing 7](#_Toc110689883)

[Potential Improvements 8](#_Toc110689884)

# Introduction

In this project, I have implemented the YOLOV3 object detection model which is a fully convolutional network (FCN) which means it only consists of convolutional layers. The YOLOV3 is a single stage detection network which means it only has the feature detection stage and then makes prediction on 3 different scales (13x13, 26x26, 52x52). The feature extractor used in YOLOV3 is called DarkNet-53. This feature makes YOLOV3 extremely fast, and its performance is also comparable to other state of the art object detection models such as Resnet. The picture below shows the YOLOV3 model architecture:



The model uses 3 anchor boxes for each scale and therefore, each cell makes 3 bounding box predictions. The concept of anchor boxes also enables the model to detect overlapping objects. However, one bounding box can only detect one object. The anchor boxes are log transformations of the original bounding boxes because using original bounding boxes leads to unstable gradients. The picture below shows the relationship between the anchor boxes and the bounding box coordinates:

Diagram, engineering drawing

Description automatically generated

The YOLOV3 model uses k-means clustering to determine the bounding box priors and then assigns them evenly to all the three scales. The following are the bounding box coordinates formulas:

Text

Description automatically generated

It is also able to do multi label classification (each bounding box can have more than 1 label). The figure below shows the output of the model for each of the bounding boxes:

Graphical user interface

Description automatically generated

# Implementation of the Model

## Architecture

The Darknet-53 architecture was implemented as it was mentioned in the research paper. The model has skip (or shortcut) connections. In these connections, 1x1 depth convolutions are done to extract features from the feature maps and then these are scaled back to the same number of channels as the input to these residual blocks to add them together. After the Darknet-53, the first scaled prediction is done at the 13x13 scale and then we have two up sampling layers each followed by the 26x26 scale and 52x52 scale predictions. The model implemented has a total of 75 convolutional layers. The output of the model at each scale is SxSx75 where S is the scale (13, 26 or 52).

The following parameters were used when building the model:

* Leaky ReLU activation function.
* Padding is SAME for all the layers.
* Batch Normalization with momentum 0.9 used before the activation function.
* Weight initialization using he\_normal initialization.
* ADAM optimizer with learning rate of 0.0001.

The anchors used for the model are the MS COCO dataset anchors mentioned in the research paper ((10×13),(16×30),(33×23),(30×61),(62×45),(59× 119),(116×90),(156×198),(373×326)).

## Loss Function

I have designed a custom loss function for each of the outputs. In the loss function, I am computing three different types of losses which are the following:

* Object Loss: This is just the object score (probability of an object existing) loss. I use binary cross entropy for computing this loss.
* Box Loss: This is the loss for the 4 coordinates (x, y, w, h). We compute this loss using the mean squared error.
* Class Loss: This is the loss for the classification of the object. I compute this using categorical cross entropy.

We do not incur loss for those bounding boxes which have an IOU of greater than 0.5 with the original bounding boxes but have not been taken to detect the object.

## Dataset Construction

I have defined a Data Generator class which inherits from keras.utils.sequence which is used to make the batches of data. I used a batch size of 32. The input which are the images are resized using albumentations library and then scaled by 255. The target of the format [object score, x, y, w, h, class label]. Here we have assigned an object score of -1 to the boxes that have an IOU of greater than 0.5 but are not being used to predict the object. This is used to filter in the loss function and not incur loss for these boxes in the Loss function. The training dataset has approximately 16500 images. The test dataset has approximately 5000 images, but we were not able to make use of that as the model was not performing very well.

## Training & Testing

I used a custom training and testing loop for training and testing the model. I saved the model weights and the optimizer (ADAM) weights after each batch.

I trained the model on two different sets. The first set has approximately 16500 images and each epoch took 3-4 hours to train the model. The second set is just 100 images which I tried to overfit using the model.

**First Data Set (16500 Images):** I trained 70 epochs on the first set but unfortunately was not able to make any predictions using that trained model. The object scores were way too low for the detections and NMS (Non-Max Suppression) therefore output an empty list for the predictions.

**Second Data Set (100 Images):** I overfit the model on the 100 images. Of course, the number of images is way too low so the model does not have a lot of features to learn about the different object so it does make the predictions but there are a lot of incorrect predictions as well. On this dataset I was able to get a Mean Average Precision of 0.157.The following pictures below show some of the outputs:

Timeline

Description automatically generatedDiagram

Description automatically generated

A picture containing text, person

Description automatically generatedA picture containing text, dog

Description automatically generated

# Potential Improvements

The following are the potential improvements which might lead to the better working of the model:

* The way the target has been formatted can be changed. I implemented it using my understanding of the research paper.
* I only used an input image size of 416x416. We should experiment with different input sizes.
* I did not use image augmentation. Image augmentation could considerably improve the results.
* I did not allow for multi label classification for each object which was done in the research paper. So, we could allow for multi label classification so that each object can have more than one label.