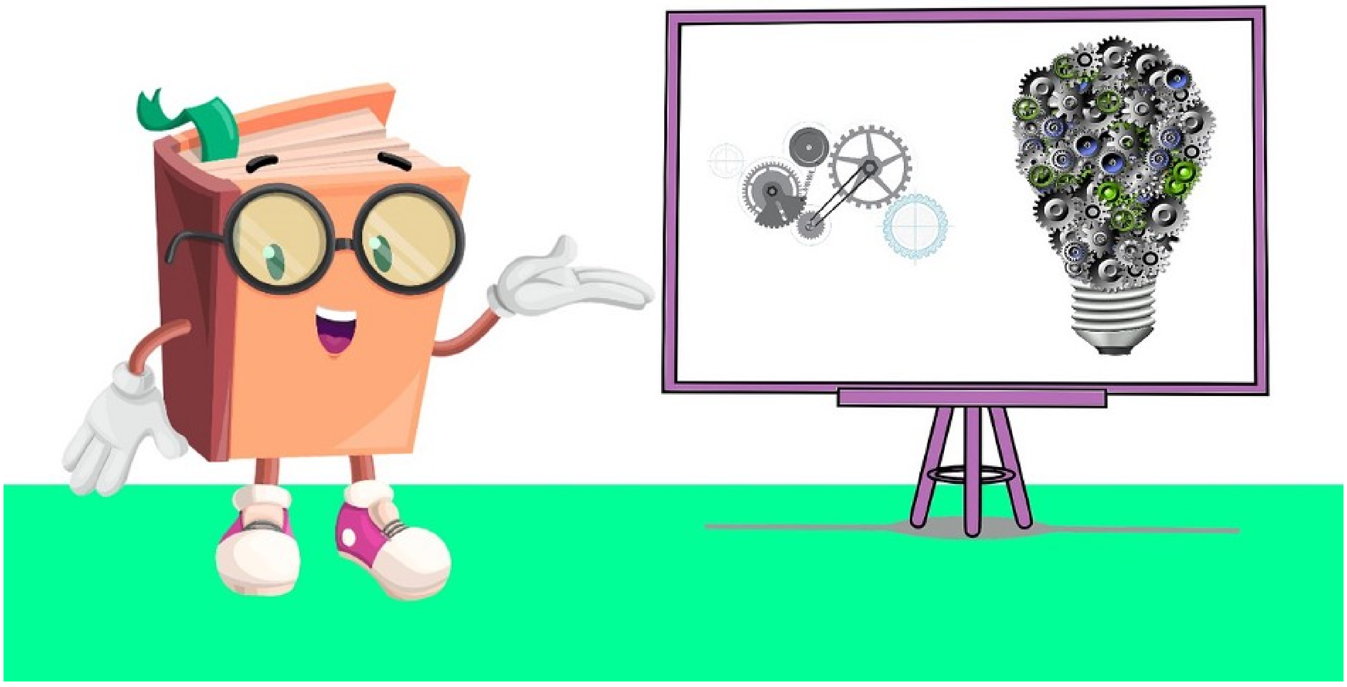
COMPUTER VISION

YOLO V5 — Explained and Demystified

YOLO V5 — Model Architecture and Technical Details Explanation

Mihir Rajput
Jul 1 · 4 min read


Source: https://pixabay.com/

From my previous article on YOLOv5, I received multiple messages and queries on how things are different in yolov5 and other related technical doubts.

Therefore, I decided to write another article to explain some technical details used in YOLOv5.

As YOLO v5 has a total of 4 versions, I will cover the ‘s’ version. But if you refer this thoroughly you will find that in other versions there are no huge changes except for the model layers/architecture and a number of parameters.

In this article, I will cover the following the most important details and aspects used in YOLOv5 implementation.

- YOLO v5 Model Architecture
- Activation Function
- Optimization Function
- Cost Function or Loss Function
- Weights, Biases, Parameters, Gradients, and Final Model Summary

> *NOTE: As YOLO v5 is still in the development phase and we are receiving updates from ultralytics frequently, in future developers may change some aspects. So this article is specifically for the initial release of YOLOv5 only. However, I will try to update/add article for subsequent releases as well.*

Let’s move to the technical discussion.

. . .

YOLO v5 Model Architecture

As YOLO v5 is a single-stage object detector, it has three important parts like any other single-stage object detector.

1. Model Backbone
2. Model Neck
3. Model Head

Model Backbone is mainly used to extract important features from the given input image. In YOLO v5 the CSP — Cross Stage Partial Networks are used as a backbone to extract rich in informative features from an input image.

CSPNet has shown significant improvement in processing time with deeper networks. Refer to the following image, for more information about CSPNet visit the Github repo.

Source: https://github.com/WongKinYiu/CrossStagePartialNetworks/blob/master/fig/cmp3.png

Model Neck is mainly used to generate feature pyramids. Feature pyramids help models to generalized well on object scaling. It helps to identify the same object with different sizes and scales.

Feature pyramids are very useful and help models to perform well on unseen data. There are other models that use different types of feature pyramid techniques like FPN, BiFPN, PANet, etc.

In YOLO v5 PANet is used for as neck to get feature pyramids. For more information on features pyramids, refer to the following link.

Understanding Feature Pyramid Networks for object detection (FPN)

Detecting objects in different scales is challenging in particular for small objects. We can use a pyramid of the same…

medium.com

The model Head is mainly used to perform the final detection part. It applied anchor boxes on features and generates final output vectors with class probabilities, objectness scores, and bounding boxes.

In YOLO v5 model head is the same as the previous YOLO V3 and V4 versions.

Additionally, I am attaching the final model architecture for YOLO v5 — a small version. You can find it here.

Activation Function

The choice of activation functions is most crucial in any deep neural network. Recently lots of activation functions have been introduced like Leaky ReLU, mish, swish, etc.

YOLO v5 authors decided to go with the Leaky ReLU and Sigmoid activation function.

In YOLO v5 the Leaky ReLU activation function is used in middle/hidden layers and the sigmoid activation function is used in the final detection layer. You can verify it here.

Optimization Function

For optimization function in YOLO v5, we have two options

1. SGD
2. Adam

In YOLO v5, the default optimization function for training is SGD.

However, you can change it to Adam by using the " — — *adam*" command-line argument.

Cost Function or Loss Function

In the YOLO family, there is a compound loss is calculated based on objectness score, class probability score, and bounding box regression score.

Ultralytics have used Binary Cross-Entropy with Logits Loss function from PyTorch for loss calculation of class probability and object score.

We also have an option to choose the Focal Loss function to calculate the loss. You can choose to train with Focal Loss by using fl_gamma hyper-parameter.

Weights, Biases, Parameters, Gradients, and Final Model Summary

To look closely at weights, biases, shapes, and parameters at each layer in the YOLOv5-small model, refer to the following information.

Source: https://gist.github.com/mihir135/969d78149b724b7684e327a1672da667

Additionally, you can also refer to the following brief summary of the YOLO v5 — small model.

```
Model Summary: 191 layers, 7.46816e+06 parameters, 7.46816e+06
gradients
```

. . .

Hopefully, this may help you to understand the YOLO v5 better. In the ending notes, I would like to thank you for reading.

Feel free to contact me for any doubts/queries/suggestions.

References:

[1] https://github.com/ultralytics/yolov5

[2] https://github.com/WongKinYiu/CrossStagePartialNetworks

Contributions:

Mayur Patel

Feel free to connect:

> LinkedIN : *https://www.linkedin.com/in/mihir-rajput/*
>
> Instagram : *https://www.instagram.com/ai_dev_/*
>
> Github : *https://github.com/mihir135/*
>
> *Thanks and Cheers!*

Artificial Intelligence Yolo Deep Learning Object Detection Data Science