## THIRD YEAR B.TECH. MINI PROJECT REPORT

*A report submitted in partial fulfillment of the requirements for the Award of a Degree of*

### BACHELOR OF TECHNOLOGY

**in**

### COMPUTER SCIENCE AND ENGINEERING

By

**Arpit Verma, Vasu Kapasiya, Mudit Loya, Priyanshu Chhabda**

### Enrollment Number: BT21CSE192, 193, 201, 219 respectively

**Object Detection Using Yolov3 with Paper Implementation**

### Course Name: Machine Learning

### Course Code: CSL 422

### 

## भारतीय सूचना प्रौद्योगिकी संस्थान, नागपुर

## INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, NAGPUR

### (An Institution of National Importance by Act of Parliament) Survey No.140,141/1 behind Br.Sheshrao Wankhede Shetkari Shakari Soot

**Girni,Nagpur, 441108"**

# TABLE OF CONTENTS

| **Sr. No.** | **Topic** | **Page No.** |
| --- | --- | --- |
|  | Introduction | 3 |
| 2. | Related Studies | 3 |
| 3. | Problem Definition | 5 |
| 4. | Model Building | 11 |
| 5. | Results and discussion | 13 |
| 6. | Conclusion | 15 |
| 7. | References | 16 |

## Introduction

### 1.1 Introduction

Welcome to our project, where we present a streamlined implementation of YOLOv3, a leading object detection model, seamlessly integrated into the PyTorch framework, with the inclusion of Darknet. Our repository stands as a testament to our commitment to delivering a minimalist yet potent solution for object detection, drawing inspiration from both the official YOLOv3 codebase and the PyTorch port by Marvis.

At the core of our endeavor lies a dedication to simplicity and efficiency. We've meticulously curated our codebase, meticulously discarding extraneous components while honing in on the essential elements that drive the effectiveness of YOLOv3. By prioritizing clarity and conciseness, we ensure that our implementation remains faithful to the principles of YOLOv3, offering uncompromising accuracy and ease of use.

Darknet, an open-source neural network framework written in C and CUDA, serves as a vital component in our project. By leveraging Darknet's capabilities, we enhance the performance and scalability of our implementation, ensuring optimal utilization of hardware resources for efficient object detection.

However, our project extends beyond mere code. Central to our mission is the empowerment of our users through education. While our current focus centers on the detection module, we are steadfast in our commitment to providing comprehensive resources for both inference and training. Our vision is to cultivate a vibrant community of learners and practitioners, fostering an environment where knowledge and expertise flourish.

In terms of technical requirements, we mandate Python 3.5, OpenCV, and PyTorch 0.4 for optimal performance and compatibility. It's crucial to adhere to these specifications, as deviations, particularly the use of PyTorch 0.3, may compromise the functionality of the detector. Our insistence on these prerequisites underscores our commitment to delivering a seamless user experience and ensuring the reliability of our implementation.

In essence, our project represents a harmonious fusion of cutting-edge technology and user-centric design. With YOLOv3 as our cornerstone, Darknet as our backbone, and PyTorch as our canvas, we invite you to embark on a journey of exploration and innovation in the captivating realm of object detection. Together, let's push the boundaries of what's possible and redefine the landscape of computer vision, one detection at a time.

### 1.2 Problem Statement

Design and implement a comprehensive object detection system capable of detecting objects in images, live web cameras, and videos using the YOLO (You Only Look Once) v3 algorithm. The system should be able to identify various objects of interest in real-time, providing accurate and timely detection across different media types. The objective is to develop a versatile and efficient object detection solution that can be deployed in diverse applications, including surveillance, video analysis, and live streaming platforms.

## Related Studies

### 2.1 Existing works

Summary of the Paper Yolo[1]:

* YOLO introduces a new approach to object detection, treating it as a regression problem and achieving real-time processing speeds.
* YOLO outperforms traditional detection systems like DPM and R-CNN, especially in generalizing to new domains.
* The approach of YOLO is to eliminate the traditional detection pipeline and focus on speed and efficiency.
* The system is designed to be fast by optimizing the entire detection process within a single neural network.
* YOLO's architecture allows for end-to-end optimization directly on detection performance.
* The system is capable of processing images in real-time at high frame rates, making it ideal for various applications.
* The loss function optimization during training emphasizes the importance of small deviations in bounding box predictions.
* The goal is to create fast and accurate algorithms for object detection, enabling applications like autonomous driving and responsive robotic systems.

Summary of Yolo9000 Object Detection System[2]:

* YOLO9000 is a real-time object detection system that can detect over 9000 object categories.
* The model YOLOv2, an improvement of YOLO, achieves state-of-the-art performance on standard detection tasks like PASCAL VOC and COCO.
* YOLO9000 is trained jointly on object detection and classification datasets, allowing it to predict detections for object classes without labeled data.
* Techniques like multi-scale training and hierarchical classification can be generalized beyond object detection tasks.
* The WordTree representation in YOLO9000 offers a detailed output space for image classification.
* YOLOv2 simplifies the network structure while maintaining accuracy, pooling ideas from past work to enhance performance.

Summary of Yolov3 for Project[3]:

* YOLOv3 is an improved version of the YOLO (You Only Look Once) object detection system, featuring design changes for better performance and accuracy.
* The new multi-scale predictions in YOLOv3 show high Average Precision (AP) for small objects but comparatively worse performance for medium and larger objects.
* YOLOv3 offers significant benefits over other detection systems in terms of speed and accuracy, making it a favorable choice for complex datasets like the Open Images Dataset.
* The system predicts bounding box dimensions and locations using various techniques, such as offset predictions and sigmoid functions.
* Researchers using computer vision, like YOLOv3, are urged to consider the ethical implications of their work and strive to mitigate any potential harm that may arise.

### 2.2 Research Gap

Based on the research papers considered for the project, it appears that while the YOLO family of object detection systems, including YOLO[1], YOLOv3[3], and YOLO9000[2], have made significant advancements in real-time object detection, there are still several research gaps that our project could address.

Firstly, despite the impressive speed and efficiency of YOLO and its variants, there may be room for further optimization, particularly in handling complex scenes with occlusions, overlapping objects, or cluttered backgrounds. Our project could explore techniques to enhance the robustness and accuracy of object detection in such scenarios.

Additionally, while YOLOv3 demonstrates improved performance over its predecessors, there are still challenges in effectively detecting medium and larger objects. Investigating strategies to address this performance gap, such as refining the network architecture or incorporating contextual information, could be a valuable contribution to the field.

Furthermore, ethical considerations in computer vision research, as highlighted in the summary of YOLOv3 for Project, are increasingly important. Our project could emphasize the importance of ethical considerations in the development and deployment of object detection systems, promoting responsible practices, and mitigating potential harm.

Overall, our project aims to build upon the existing works in the field of object detection by addressing these research gaps and pushing the boundaries of speed, accuracy, and ethical considerations in real-time object detection systems.

## 

## 

## 3. Problem Definition

### 3.1 Problem Statement

Design and implement a comprehensive object detection system capable of detecting objects in images, live web cameras, and videos using the YOLO (You Only Look Once) v3 algorithm. The system should have the capability to identify various objects of interest in real-time, providing accurate and timely detection across different media types. The objective is to develop a versatile and efficient object detection solution that can be deployed in diverse applications, including surveillance, video analysis, and live streaming platforms.

**Key Objectives:**

* Develop a deep learning model based on the YOLO v3 architecture for object detection in images, live web cameras, and videos.
* Train the model using a diverse dataset containing annotated images, live camera feeds, and video clips, covering a wide range of object categories and environmental conditions.
* Implement real-time processing capabilities to enable the model to perform object detection on live web cameras and video streams with minimal latency.
* Optimize the model for accuracy, speed, and efficiency, ensuring reliable performance across different media types and processing environments.
* Evaluate the performance of the trained model on benchmark datasets and real-world scenarios, assessing its detection accuracy, precision, recall, and processing speed.
* Fine-tune the model and hyperparameters as necessary to improve detection performance and address any shortcomings identified during the evaluation.
* Develop a user-friendly interface or application for accessing and utilizing the object detection system, allowing users to perform detection tasks on images, live camera feeds, and videos seamlessly.
* Provide documentation and guidelines for deploying and using the object detection system, including instructions for integrating it into various software applications, platforms, or devices.

**Deliverables:**

* A trained object detection model based on YOLO v3 can accurately detect objects in images, live web cameras, and videos.
* Real-time processing capabilities implemented for live web cameras and video streams, enabling efficient object detection in real-time scenarios.
* Evaluation metrics and performance analysis reports demonstrating the detection performance and processing speed of the model across different media types and environments.
* Source code and documentation detailing the implementation of the object detection system, including instructions for model training, evaluation, and deployment.
* User interface or application for accessing and utilizing the object detection system, with intuitive controls for performing detection tasks on various media types.
* User guidelines and instructions for deploying and utilizing the system in real-world applications, including integration with existing software or hardware platforms.

**Success Criteria:**

* Achieving high detection accuracy and processing speed across different media types, with precision, recall, and F1-score meeting predefined thresholds.
* Demonstrating real-time object detection capabilities on live web cameras and video streams, with minimal latency and efficient resource utilization.
* Seamless integration of the object detection system into practical use cases, with clear documentation and guidelines for users.
* Positive feedback from users and stakeholders on the usability, reliability, and effectiveness of the object detection solution in real-world scenarios.

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

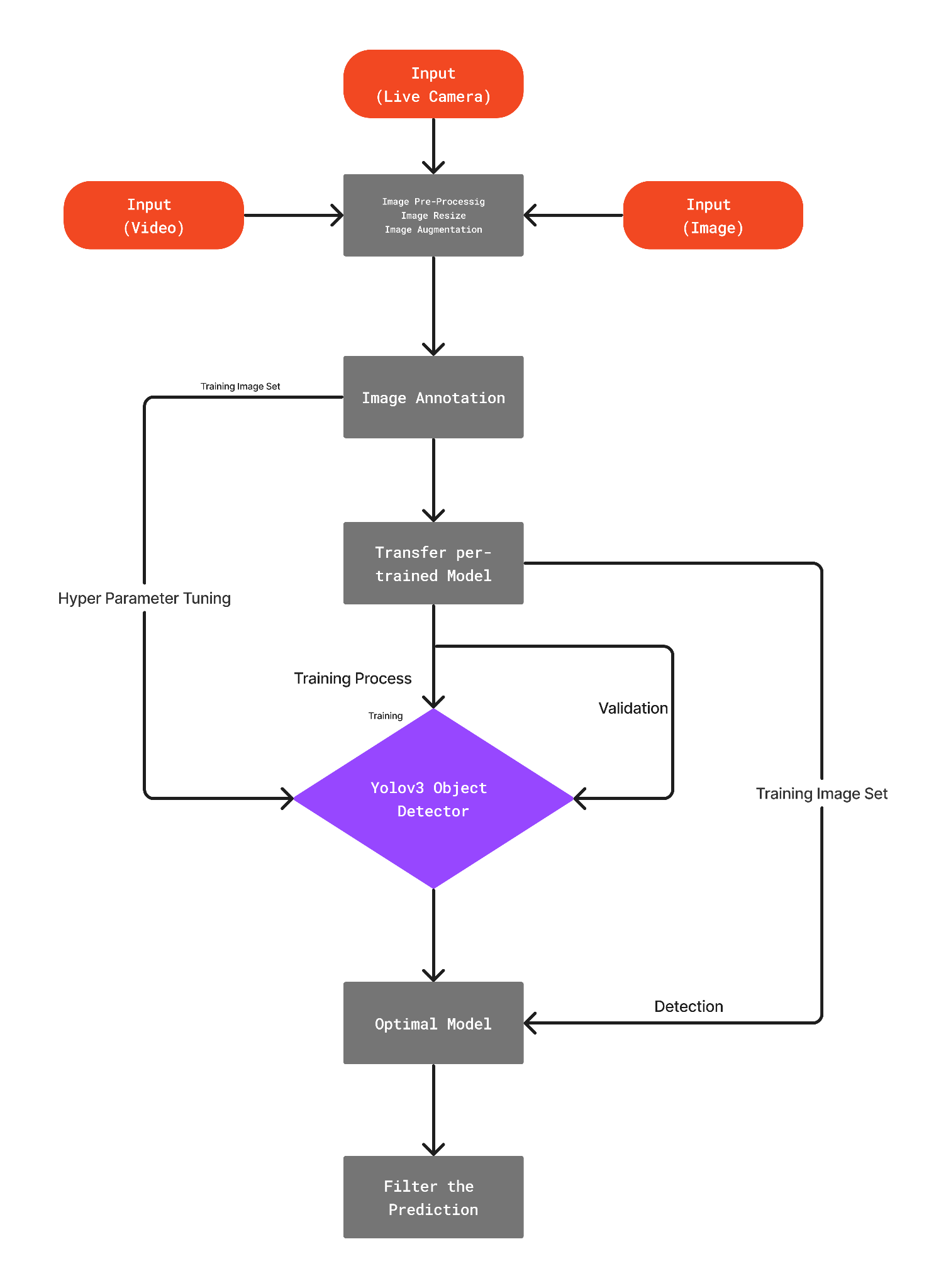
### 

### 

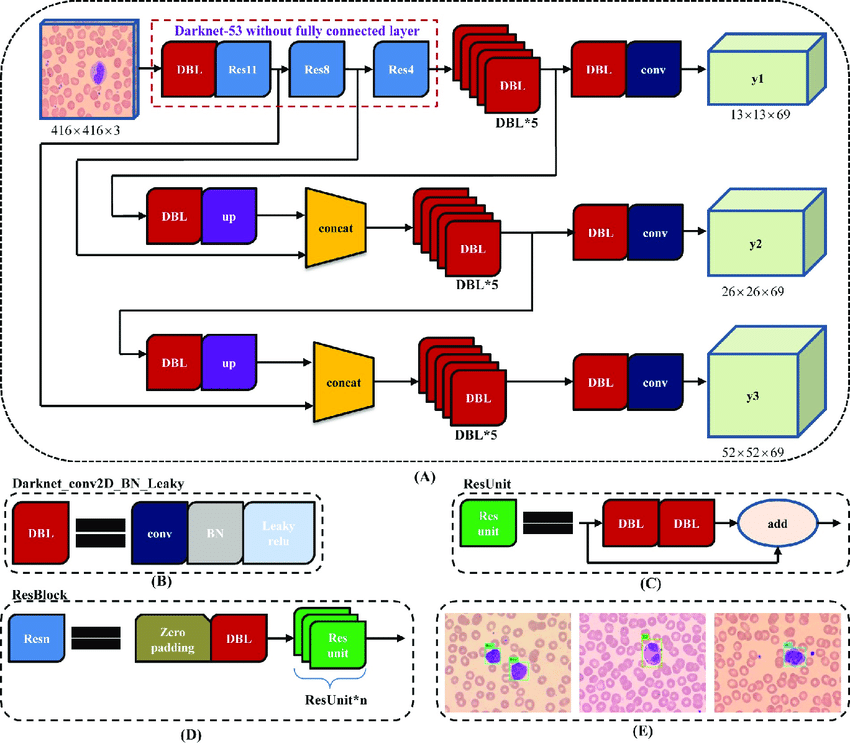
### 

### 

### 3.2 Flowchart/Block Diagrams



**Fig: YOLO-V3 complete training diagram**



**Fig: YOLOv3 architecture. (A) YOLOv3 pipeline with input image size 416×416 and 3 types of feature map (13×13×69, 26×26×69 and 52×52×69) as output; (B) the basic element of YOLOv3, Darknet\_conv2D\_BN\_Leaky ("DBL" for short), is composed of one convolution layer, one batch normalization layer and one leaky relu layer.; (C) two "DBL" structures following with one "add" layer lead to the residual-like unit ("ResUnit" for short); (D) several "ResUnit" with one zero padding layer and "DBL" structure forward generates a residual-like block, "ResBlock" for short, which is the module element of Darknet-53; (E) some detection results of peripheral leukocyte using YOLOv3 approach, resize the 732×574 images to 416×416 size as input.**

### 

Fig:

### 

### 3.3 Dataset Description

* The MS COCO (Microsoft Common Objects in Context) dataset is a large-scale object detection, segmentation, key-point detection, and captioning dataset. The dataset consists of 328K images.
* Splits: The first version of the MS COCO dataset was released in 2014. It contains 164K images split into training (83K), validation (41K) and test (41K) sets. In 2015 additional test set of 81K images was released, including all the previous test images and 40K new images.
* Based on community feedback, in 2017 the training/validation split was changed from 83K/41K to 118K/5K. The new split uses the same images and annotations. The 2017 test set is a subset of 41K images of the 2015 test set. Additionally, the 2017 release contains a new unannotated dataset of 123K images.

**Annotations:**

The dataset has annotations for object detection: bounding boxes and per-instance segmentation masks with 80 object categories, captioning: natural language descriptions of the images (see MS COCO Captions), key points detection: containing more than 200,000 images and 250,000 person instances labeled with key points (17 possible key points, such as left eye, nose, right hip, right ankle), stuff image segmentation – per-pixel segmentation masks with 91 stuff categories, such as grass, wall, sky (see MS COCO Stuff), panoptic: full scene segmentation, with 80 thing categories (such as a person, bicycle, elephant) and a subset of 91 stuff categories (grass, sky, road), dense pose: more than 39,000 images and 56,000 person instances labeled with DensePose annotations – each labeled person is annotated with an instance ID and a mapping between image pixels that belong to that person's body and a template 3D model. The annotations are publicly available only for training and validation images.

### 3.4 Dataset Source and Attribute Information

* Source: <https://cocodataset.org/#download>[6]
* Attribute Information

Info: general information about the dataset, such as version number, date created, and contributor information.

Licenses: information about the licenses for the images in the dataset

Images: a list of all the images in the dataset, including the file path, width, height, and other metadata

Annotations: a list of all the object annotations for each image, including the object category, bounding box coordinates, and segmentation masks (if available)

Categories: a list of all the dataset object categories, including each category's name and ID

### 3.5 Experimental setup

**Google Colab Environment:**

* Hardware Acceleration: For model training and experimentation, Google Colab's environment was utilized, which provides access to GPU accelerators such as "Tesla K80" or "Tesla T4". This enabled faster model training compared to traditional CPU-based environments.

**Library used:**

* **OpenCV (Open Source Computer Vision Library)**: OpenCV is a widely-used open-source library for computer vision tasks, offering functions for image and video processing, object detection, and feature extraction. It provides efficient and comprehensive tools for handling various aspects of computer vision applications.
* **PyTorch**: PyTorch is a popular deep learning framework known for its flexibility, dynamic computation graphs, and GPU acceleration capabilities. It simplifies the process of building and training neural networks, making it a preferred choice for researchers and practitioners in the machine learning community.
* **Darknet**: Darknet is a lightweight neural network framework written in C and CUDA, optimized for training and deploying deep neural networks. It provides efficient implementations of key operations required for deep learning tasks like object detection and image classification, enabling high-performance inference on various hardware platforms.

**Software Integration**:

Colab seamlessly integrates with Google Drive, allowing for easy access to datasets and model checkpoints stored in the cloud.

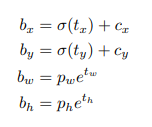
**Collaborative Workflow**:

Colab supports collaborative editing and real-time collaboration, facilitating teamwork and knowledge sharing among project members.

## 4. Model Building

**Bounding Box Prediction[2][3]:**

The network predicts 4 coordinates for each bounding box, tx, ty, tw, th. If the cell is offset from the top left corner of the image by (cx, cy) and the bounding box prior has width and height pw, ph, then the predictions correspond to:



During training, we use the sum of squared error loss. If the ground truth for some coordinate prediction is tˆ \* our gradient is the ground truth value (computed from the ground truth box) minus our prediction: tˆ \* − t\*. This ground truth value can be easily computed by inverting the equations above. YOLOv3 predicts an objectness score for each bounding box using logistic regression. This should be 1 if the bounding box prior overlaps a ground truth object by more than any other bounding box. If the bounding box prior is not the best but does overlap a ground truth object by more than some threshold we ignore the prediction, following.

**Class Prediction[3]:**

Each box predicts the classes the bounding box may contain using multilabel classification. We do not use a softmax as we have found it is unnecessary for good performance, instead, we simply use independent logistic classifiers. During training, we use binary cross-entropy loss for the class predictions.

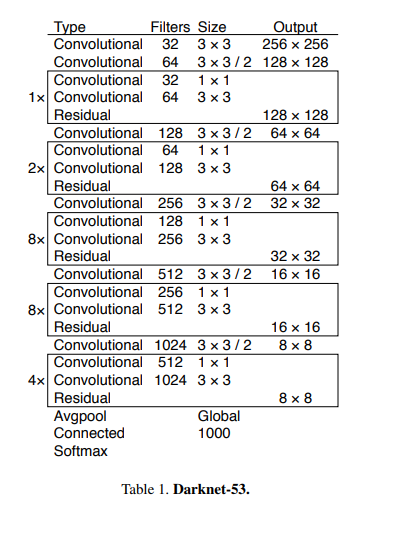
This formulation helps when we move to more complex domains like the Open Images Dataset. In this dataset, there are many overlapping labels (i.e. Woman and Person). Using a softmax imposes the assumption that each box has exactly one class which is often not the case. A multilabel approach better models the data.

**Predictions Across Scales[3]:**

YOLOv3 predicts boxes at 3 different scales. Our system extracts features from those scales using a similar concept to feature pyramid networks. From our base feature extractor, we add several convolutional layers. The last of these predicts a 3-D tensor encoding bounding box, objectness, and class predictions. In our experiments with COCO [6] we predict 3 boxes at each scale so the tensor is N × N × [3 ∗ (4 + 1 + 80)] for the 4 bounding box offsets, 1 objectness prediction, and 80 class predictions.

**Feature Extractor[3]:**

We use a new network for performing feature extraction. Our new network is a hybrid approach between the network used in YOLOv2, Darknet-19, and that newfangled residual network stuff. Our network uses successive 3 × 3 and 1 × 1 convolutional layers but now has some shortcut connections as well and is significantly larger. It has 53 convolutional layers so we call it.... wait for it..... Darknet-53!

****

**Fig: Darknet 53 Architecture from YOLO V-3[3]**

Each network is trained with identical settings and tested at 256×256, single crop accuracy. Run times are measured on a Titan X at 256 × 256. Thus Darknet-53 performs on par with state-of-the-art classifiers but with fewer floating point operations and more speed. Darknet-53 is better than ResNet-101 and 1.5× faster. Darknet-53 has a similar performance to ResNet-152 and is 2× faster.

**Training[3]**:

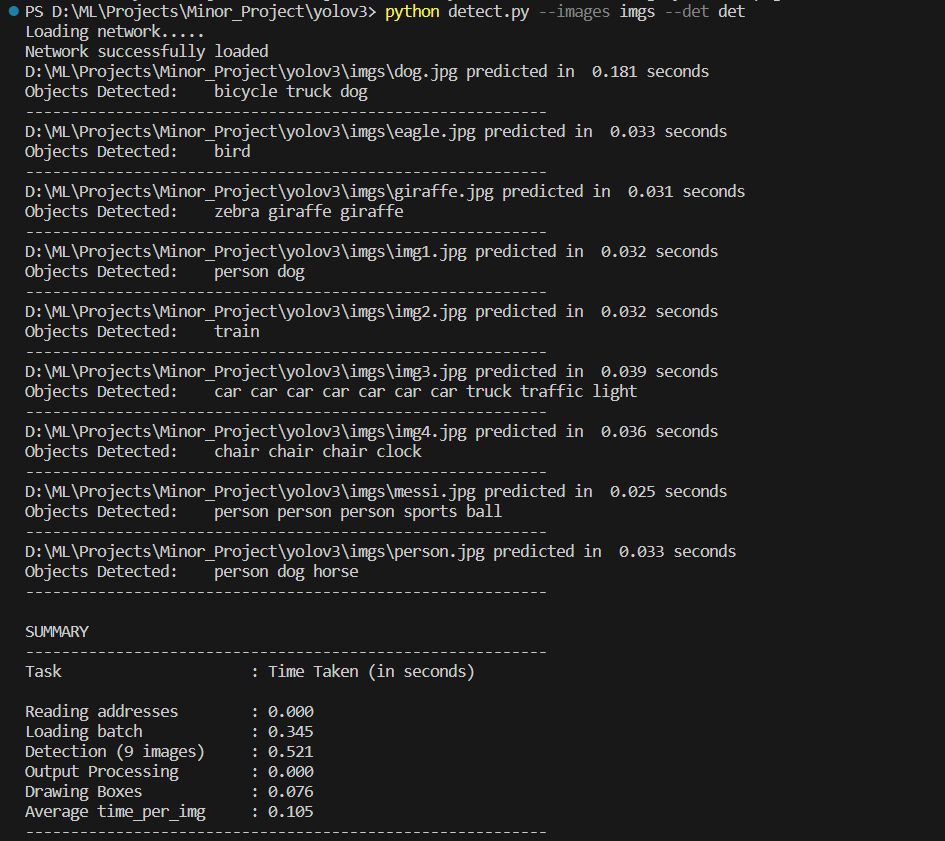
We still train on full images with no hard negative mining or any of that stuff. We use multi-scale training, lots of data augmentation, batch normalization, and all the standard stuff. We use the Darknet neural network framework for training and testing.

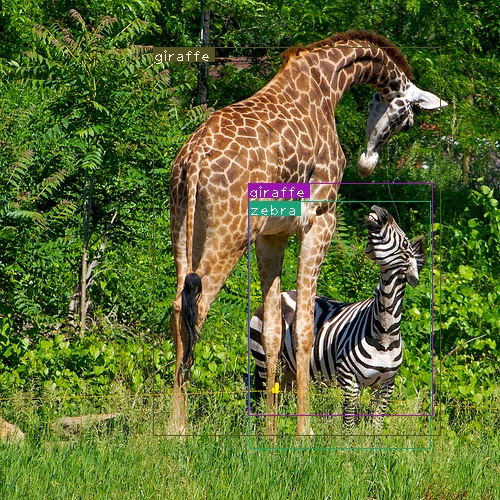
| **Model** | **Backbone** | **mAP** | **mAP-50** | **Time** |
| --- | --- | --- | --- | --- |
| Yolov3-416 | Darknet 53 | 31.0 | 55.3 | 29 |

## 

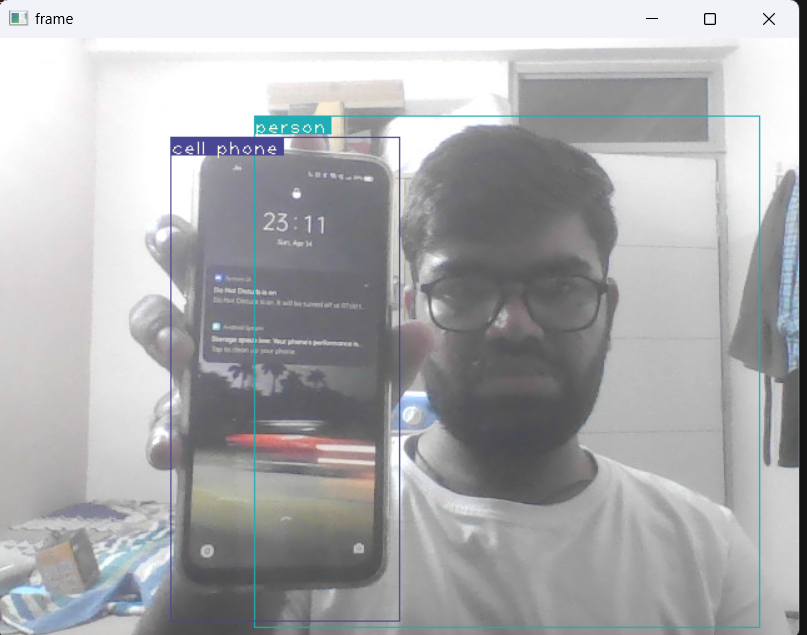
## 5. Results and discussion

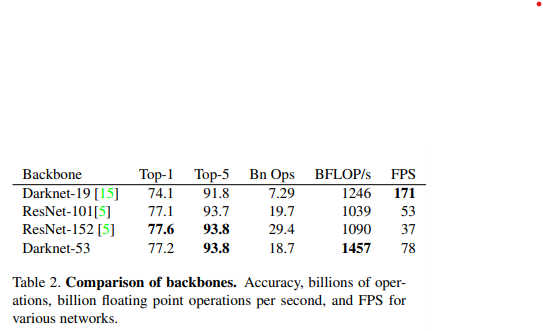
**Image Detection:**





**Live-Cam Demonstration:**

****

****

**Fig: Comparison of backbones from YOLO V-3[3]**

## 6. Conclusion

YOLOv3 is a good detector. It’s fast, it’s accurate. It’s not as great on the COCO average AP between .5 and .95 IOU metric. But it’s very good on the old detection metric of .5 IOU. Why did we switch metrics anyway? The original COCO paper just has this cryptic sentence: “A full discussion of evaluation metrics will be added once the evaluation server is complete”. Russakovsky et al report that humans have a hard time distinguishing an IOU of .3 from .5! “Training humans to visually inspect a bounding box with an IOU of 0.3 and distinguish it from one with an IOU of 0.5 is surprisingly difficult.”

If humans have a hard time telling the difference, how much does it matter? But maybe a better question is: “What are we going to do with these detectors now that we have them?” A lot of the people doing this research are at Google and Facebook. I guess at least we know the technology is in good hands and definitely won’t be used to harvest your personal information and sell it to.... wait, you’re saying that’s exactly what it will be used for?? Oh. Well the other people heavily funding vision research are the military and they’ve never done anything horrible like killing lots of people with new technology oh wait.....1 I have a lot of hope that most of the people using computer vision are just doing happy, good stuff with it, like counting the number of zebras in a national park, or tracking their cat as it wanders around their house. But computer vision is already being put to questionable use and as researchers we have a responsibility to at least consider the harm our work might be doing and think of ways to mitigate it. We owe the world that much.

## 7. References

[1] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi <http://pjreddie.com/yolo/> You Only Look Once: Unified, Real-Time Object Detection.

[2] Joseph Redmon, Ali Farhadi <http://pjreddie.com/yolo9000/> YOLO9000: Better, Faster, Stronger.

[3] Joseph Redmon, Ali Farhadi YOLOv3: An Incremental Improvement.

[4] Zhong-Qiu Zhao, Peng Zheng, Shou-tao-Xu Xindong Wu Object Detection with Deep Learning: A Review.

[5] Tausif Diwan & G. Anirudh & Jitendra V. Tembhurne Object detection using YOLO: challenges, architectural successors, datasets and applications.

[6] Tsung-Yi Lin Michael Maire Serge Belongie Lubomir Bourdev Ross Girshick James Hays Pietro Perona Deva Ramanan C. Lawrence Zitnick Piotr Dollar Microsoft COCO: Common Objects in Context [1405.0312.pdf (arxiv.org)](https://arxiv.org/pdf/1405.0312.pdf)

### 