**Chapter 6**

**A Deep Learning-based Food detection and Classification**

Bhan Singh1a, Divyanshu1b, Mayur Kashyap1c, Himanshu Gupta1d, Om Prakash Verma1e\*

Department of Instrumentation and Control Engineering, Dr. B R Ambedkar National Institute of Technology Jalandhar, India

Email addresses: bhans1709@gmail.com1a (B.S.), [divyanshu102k@gmail.com1b (D.),](about:blank)

mayurkashyap30@gmail.com1c (M.K.), [guptah.nitj@gmail.com1d](mailto:guptah.nitj@gmail.com1d) (H.G.)

\*Corresponding author email address: vermaop@nitj.ac.in1e(O.P.V.)

**Abstract**

Nutrition is a basic necessity and makes a necessary part of the survival and development of every species. Conventional nutrition level measuring methods result in various drawbacks like complexity and time-consuming. This can be overcome by Convolutional Neural Network-based food detection techniques. Food-based industries require inventory management and sensor-based control systems as counters. Control systems that are emitting radiations and suffering from component decay, cause various losses as well as harmful effects. Furthermore, these are often not cheap. However, a simple Deep Learning-based food detector can help solve the aforementioned issues. Therefore, the primary target of the proposed work is to endorse the accuracy and efficiency of three deep learning-based object detection models namely YOLOv3, YOLOv4, and SSD based on mAP, and draw useful conclusions from the results. To carry out the detection process, the dataset of fast food items is chosen which includes various types of commonly consumed food items. Due to the large pool of fast food items, 10 categories have been selected to ease the research. The research yields a 74.55% mAP for YOLOv3 and 71.26% mAP for YOLOv4, compared to 13.59% mAP for an old model of SSD.

*Keywords*: Deep Learning, Food Counter, Food Detection, Nutrition, SSD, YOLO

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Abbreviations** | | | | | |
| AP | : | Average Precision | PANet | : | Path Aggregation Network |
| BoF | : | Bag of Freebies | PASCAL VOC | : | Pattern Analysis, Statistical Modelling and  Computational Learning Visual Object Classes |
| BoS | : | Bag of Specials | RCNN | : | Region-based Convolutional Neural Networks |
| CIoU | : | Complete Intersection over Union | SGD | : | Stochastic Gradient Descent |
| CNN | : | Convolutional Neural Network | SPM | : | Spatial Pyramid Match |
| CSP | : | Cross Stage Partial connections | SPP | : | Spatial Pyramid Pooling |
| F1-Score | : | Harmonic Mean of Precision and Recall | SSD | : | Single Shot Detection |
| FN | : | False Negative | TN | : | True Negative |
| FP | : | False Positive | TP | : | True Positive |
| IoU | : | Intersection Over union | VGG | : | Visual Geometry Group |
| mAP | : | mean Average Precision | YOLO | : | You Only Look Once |
| NMS | : | Non Maximal Suppression |  |  |  |

# Introduction

Nutrition is an essential part of human health. It not only helps in reducing the risk of non-communicable diseases and improves the immune system but is also directly associated with safer pregnancy and longevity. People with adequate nutrition tend to be more productive creating better chances for progress and breaking cycles of hunger and poverty. Further, malnutrition and its ally are considered as the most prominent cause particularly, in children below 5 years which resulted in 45% of total deaths [1]. Keeping track of nutrition intake has become a necessary measure to prevent the negative effects of malnutrition. Previous methods of nutrition check mostly focused on clinical and growth change methods which are fairly complex, time-consuming, and usually require special equipment handled by professionals. Lately, AI-based object detection methods have gained popularity and are being used for similar objectives. Standard nutritional values can be displayed for the detected food and an application can be developed.

Another food-related issue often faced is related to inventory. Food chain industries like restaurants, street vendors, etc. need to keep track of different food items in storage, after preparing, and for delivery or takeout purposes. Assigning people to keep track of such tasks results in errors and is a major cause of losses in the big run. Conventionally, sensor-based control systems have been used to replace people. Such control systems consist of simple IR, UV, or electric pulse-based control systems that can be scaled to the required size. These methods often result in the emission of radiation or component deterioration which can be harmful to food items, especially cooked food [2]. These methods are highly expensive. A simple camera-based application running on a food detecting algorithm can solve the aforementioned issues.

Deep Learning (DL) in artificial intelligence is a technique imitating the functions of the human brain in detection, and data processing. Neural networks came into play in the 1940s. Neural networks aim at solving problems related to learning ethically and efficiently. DL models have been employed to solve the problems related to prediction and complex object detection [3-6]. Convolutional neural networks (CNN) are frequently used in such models. This set of rules takes input snapshots and assigns learnable weights and biases for diverse targets in a photo and makes them different from each other. Furthermore, a low- resolution image with low complexity may be used with convolutional neural networks giving an efficient output. These strategies assign a bounding container over the items and associate the perfect item elegance for every container. It can be performed using various DL algorithms like Region-based CNN, YOLOv3 [7], YOLOv4 [8], and Single-shot detector (SSD) [9]. Therefore, in this work three DL-based object detection models (YOLOv3, YOLOv4, and SSD) have been employed to develop a food detection system.

Conclusively, the key contributions of the present proposed work can be summarized as follows:

* Trained models for various food detection applications like nutrition check and food item counter.
* Inferences on YOLOv3, YOLOv4, and SSD that deflect from the norm using an imbalanced dataset.

The rest of the paper is divided mainly into 5 sections. Covering Literature Review in Section 6.2, Section 6.3 introduces the theory of YOLOv3, YOLOv4, and SSD. The dataset operated upon for training and testing is briefly presented in the methodology/experiments section (Section 6.4). It explains the software and hardware used for training the models along with a skimp description of the implementation of all 3 models. Section 6.5 presents and discusses the results obtained from the experiments. Lastly, Section 6.6 gives concluding remarks for the paper and its future scope.

# 6.2 Literature Review

# Redmon *et al.* introduced a new system to detect an object in May 2016 [10]. In his paper, the system needs to look only once (YOLO) at an image to predict the class and position of an object. The architecture of YOLO is shown in Fig. 6.1. Before YOLO, system took a classifier to detect an object with different positions and scales in a test image. YOLO algorithm achieved a 57.9% precision on the PASCAL VOC 2012. In Dec 2016, the same authors came up with an improved YOLO algorithm named YOLOv2. This algorithm outperformed their previous work. YOLOv2 achieved a precision of 73.4% on the same PASCAL VOC 2012 dataset [11]. YOLOv3 [7], published in April 2018, achieved 57.5% AP (50) in 51 ms whereas, RetinaNet achieved 57.9% AP (50) in 198ms, similar performance but 3.8 times faster. YOLOv3 was 3 times faster than SSD with similar performance in terms of efficiency. YOLOv4 [8], introduced by Alexey in April 2020, was efficient than YOLOv3 by 12%. Its Accuracy was 10% better than YOLOv3. SSD, [9] published in Dec 2016 by Wei Liu, uses a single neural network for object detection in snapshots. Guneet in her paper [12] identifies that SSD performs well on the dataset of Aircraft images which is having a wide range, scale, and oriented objects in an image. SSD gives 0.916 accuracy on the validation dataset of Aircraft images. It outperforms YOLOv3 (0.88) and RCNN (0.63). Asjad *et al.* in their paper [13] compares YOLOv3 and YOLOv4 on vehicle dataset. It shows that both these algorithms are fast but YOLOv4 gives 0.81 precision and YOLOv3 gives 0.76 when training of final weight on vehicle dataset is finished. Similarly, Gupta *et al.* employ Faster R-CNN, SSD, YOLOv3, and YOLOv4 for vehicle detection using an aerial image dataset and identify that even with a bird-eyed view of the dataset, YOLOv4 outperforms others [14].

# 6.3 Theory

The primary aim for any object detection algorithms is to be able to recognize a wide variety of objects with acceptable accuracy at a reasonable speed. Different models' performance should be measured to achieve the stated goals. The following sections study YOLOv3, YOLOv4, and SSD models along with their architectures.

## 6.3.1 YOLOv3

YOLOv3 employs dimension clusters as anchor boxes to predict bounding boxes. Utilizing logistic regression, it predicts the objectness score for each bounding box, which is closer to 1 if more overlapping is observed between ground truth and bounding box prior compared to any other bounding box priors. Unlike classification or coordinates, objectness is lost if the bounding box prior is not allocated to the ground truth. The entire topmost feature map is employed for forecasting both confidences for multiple categories as well as bounding boxes. With the increase in Intersection over Union (IoU) threshold, the performance of YOLOv3 decreases significantly [7]. Further, during training, it utilizes the Binary cross-entropy loss for making class predictions. In complex domains like Open Image Dataset (OID), this loss and independent logistic classifiers are really helpful. Data augmentation, multi-scale training, and batch normalization are used for training [15]. YOLOv3 adds a fully connected layer for prediction, unlike SSD which uses a convolutional predictor with multiple aspect ratios.

**Architecture**

Network in YOLOv3 takes a hybrid approach between YOLOv2 network and Darknet19 [16]. It is structured of successive 3×3 and 1×1 convolutional layers totaling 53 convolutional layers along with some skip connections thus called Darknet53 as shown in Fig. 6.1. It predicts 4 coordinates *tx*, *ty*, *tw*, and *tz* for every bounding box. The bounding box predictions are calculated according to the equations (6.1)-(6.5) respectively.

*bx = σ(tx) + cx* (6.1)

*by = σ(ty) + cy* (6.2)

*bz = σ(tz) + cz* (6.3)

*bw =* (6.4)

*bh =* (6.5)

where, *pw* and *ph* indicate the width and height of the bounding box priorwhereas, *cx*, and *cy* represents the offsets for a cell in the top-left coordinate of the image. Ground truth values can be evaluated by backtracking the equations (6.1)-(6.5).

The last convolutional layer predicts a 3D tensor for encoding box, class, along objectness. Feature layers are concatenated with feature maps from previous layers. Therefore, it is able to provide rich semantic and spatial information from upsampled features and earlier feature maps respectively [7]. Bounding box prior is determined using *k*-means clustering.

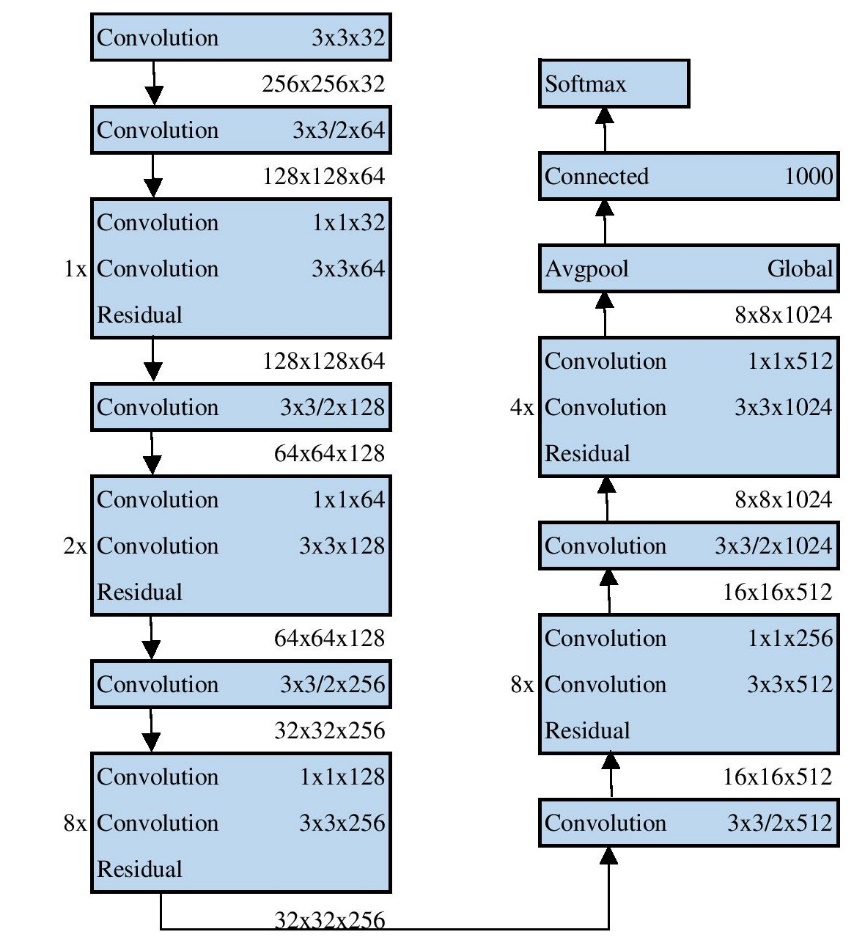


Fig 6.1. YOLOv3 Architecture

## 6.3.2 YOLOv4

YOLOv4 achieves a fast operating, real-time high object detector with the introduction of two bags i.e. bag of freebies and bag of specials.

Bag of Freebies (BoF): Methods to receive better accuracy by changing training strategy without increasing inference cost. For the backbone of YOLOv4, freebies are cut mix, mosaic data augmentation, drop block regularization, and class label smoothing. For detectors, freebies include Complete Intersection over Union (CIoU) loss and self-adversarial training [17, 18].In addition to data augmentation, a bag of freebies solves the problem of semantic distribution bias in the dataset.

Bag of specials (BoS): Methods that enhance the object detection accuracy significantly at cost of a small increase in inference cost. For backbone, specials are mish activation and cross-stage partial connection. For detector, bags of specials have Spatial Pyramid Pooling (SPP) and Path Aggregation Network (PANet) [19]. Knowledge distillation is used for better smoothing of the label. Class label smoothing makes the model more robust [8].

IoU loss is directly used to find the coordinate value of each corner of the bounding box. Spatial Pyramid Match (SPM) splits the feature map into d\*d identical blocks with d=1, 2… forms a spatial pyramid. k\*k max-pooling increases the susceptive field of the feature map as it is relatively large [20]. Activation statistics from 4 contrasting figures are evaluated on every individual layer using batch normalization diminishing the wanting of a larger mini-batch size. The addition of BoF and BoS training strategies makes detector performance independent of mini-batch size.

**Architecture**

YOLOv4 architecture is divided into 3 parts as shown in Fig 6.2 i.e. backbone, head, and some layers inserted between them that are called the neck.

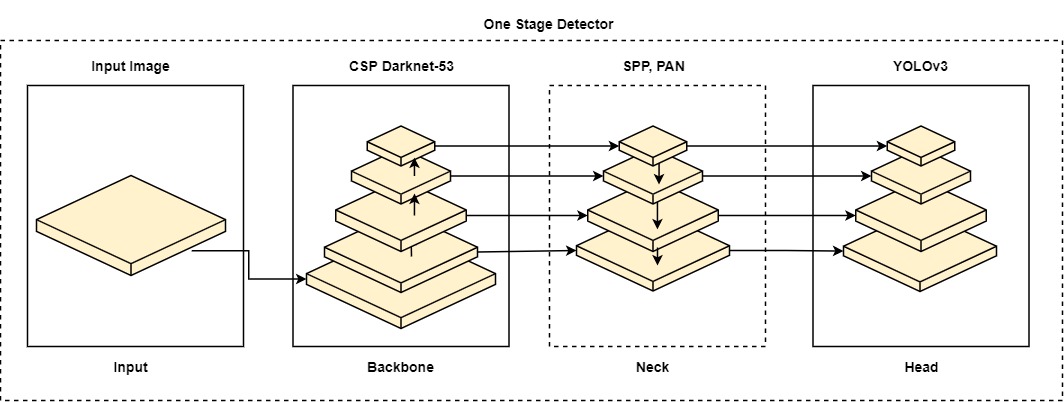


Fig 6.2. YOLOv4 Architecture

It uses CSPDarknet53 as its backbone, containing 29 convolutional layers 3×3, 725×725 receptive fields, and 27.6 million parameters. The influence of receptive fields with different sizes depends upon the object size i.e. entire viewing of the object and network size i.e. context around the object. SPP block is added over CSPDarknet53 to increase the receptive field significantly. PANet is used as neck consisting path aggregation from different backbone levels from different level detectors and the head is structured by implementing YOLOv3 (anchor-based).

## 6.3.3 SSD

It discretizes the output area of the bounding box, utilizing several combinations of disparate aspect ratios and scales as per feature map location into default boxes. These default boxes are almost similar to the anchor boxes as employed by earlier techniques except that now, they are applied on many feature maps with different resolutions. It is centered on a feedforward convolutional network that provides a set number of bounding boxes and scores for the attendance of an object in the class accompanied by non-maximal suppression. This network mixes forecasts from a myriad of feature maps to automatically handle objects of different sizes. Lower layers of the network give fine details of the detected object. Feature maps from these layers enhance the semantic segmentation value. It is more sensitive to bounding box which causes poor performance in detecting small objects compared to large objects [9].

**Architecture**

VGG-16 network is used as the base in SSD 300. The default boxes combine the feature maps in such a way so that the relative position of each box with the corresponding cell remains unchanged. These generated default boxes are illustrated in Fig 6.3 for better visualization. Further, for every box out of *M* from a given locality, it estimates the *c* class score and 4 offsets relative to the default box [21]. The gross objective loss can be computed by equation 6.6 which is the weighted aggregate of localization loss and confidence loss.

(6.6)

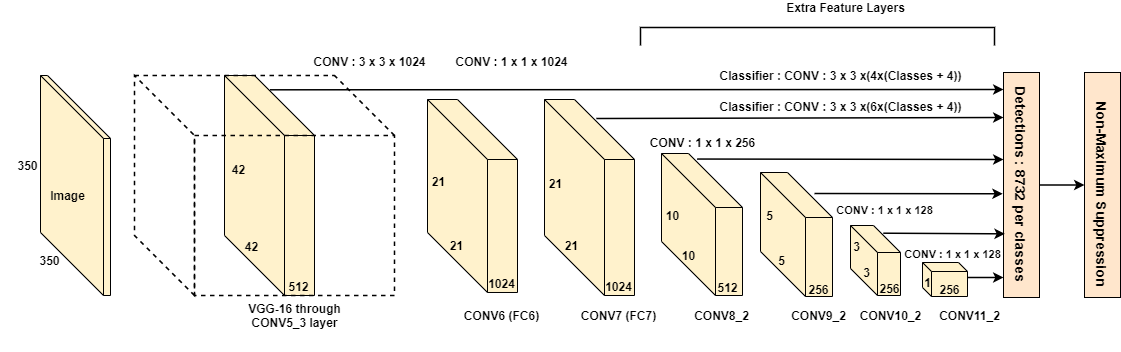


Fig 6.3. SSD Architecture

If N (number of matched default boxes) =0 then, the loss will also be 0. After matching of default boxes, most of them being negative, they are sorted using the highest confidence loss for each with the ratio at most being 3:1. This leads to faster optimization and stable training. Top 200 detection per image can be kept after applying non-maximal suppression with Jaccard overlap 0.45 class [9]. Non-Maximal Suppression (NMS) is a post-processing method used to filter out that bounding box which predicts the same object badly but retains those with a higher response.

# 6.4 Methodology/Experiments

This experiment is an attempt to train an object detection model based on single-shot detectors with decent mAP that can be used for real-world applications. The following sections contain details about the dataset, its preparation, and implementation of three single-state detection algorithms namely YOLOv3, YOLOv4, and SSD.

## 6.4.1 Dataset

The dataset consists of images from open source Open Image Dataset (OID) [22] by Google containing masks for segmentation, labels for classification along bounding box annotations for more than 600 categories. Given the large size of the dataset, the OIDv4 Toolkit for downloading a smaller number of images from 10 classes is used for the experiment. Dataset’s uneven distribution in Table 6.1 is due to less number of images of particular classes like a burrito and hot dog available on Google Open Image Dataset and a large number of images of fast food, pizza, and pasta. Moreover, this distribution helps us to analyze the model performance when the dataset is rich and also when it is weak. Fig 6.4 represents the graphical distribution of the dataset.

Table 6.1 Dataset Division for Each Class

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Class** | Bread | Burrito | Candy | Fast Food | French Fries | Hamburger | Hot Dog | Pasta | Pizza | Snack | **Total** |
| **Training** | 508 | 46 | 210 | 1064 | 331 | 372 | 98 | 268 | 404 | 750 | 4051 |
| **Validation** | 197 | 15 | 63 | 250 | 66 | 65 | 8 | 113 | 76 | 250 | 1103 |
| **Test** | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 500 |
| **Total** | 755 | 111 | 323 | 1364 | 447 | 487 | 156 | 431 | 530 | 1050 | 5654 |

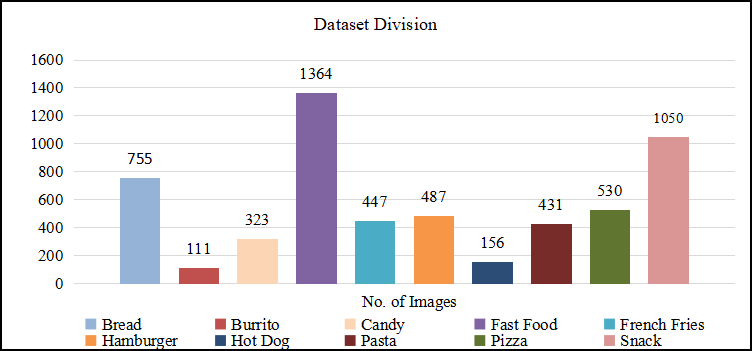


Fig 6.4. Dataset Division

## 6.4.2 Data Augmentation

As the DL models are data-driven and the images are not in an adequate amount, training with these images can result in low accuracy of the model. So, to increase the volume of images, a data augmentation technique is used. For each image, six transformations like horizontal flip, vertical flip, horizontal and vertical flip, blur, hue, and rotation are applied using CloDSA toolkit [23] for bringing in more variations in the training dataset, hence increasing the robustness of the detector. The summary of images before and after the augmentation is shown in Table 6.2.

Table 6.2 Dataset Before and After Augmentation

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Class** | Bread | Burrito | Candy | Fast Food | French Fries | Hamburger | Hot Dog | Pasta | Pizza | Snack | **Total** |
| **Before Aug.** | 755 | 111 | 323 | 1364 | 447 | 487 | 156 | 431 | 530 | 1050 | 5654 |
| **After Aug.** | 3556 | 322 | 1470 | 7448 | 2312 | 2604 | 686 | 1876 | 2828 | 5250 | 28352 |

The fast food dataset has the largest number of images (7448) while Burrito has the lowest number of images (322) as shown in Table 6.2.

## 6.4.3 Implementation

The dataset is split into training, validation, and testing datasets each occupying 71.65% (20314), 19.51% (5531), and 8.84% (2507) of 28352 images in total, respectively. All the models have been trained end to end with weights of corresponding backbones trained on ImageNet used as initial weights.

YOLOv3: To train our YOLOv3 model, the text annotations with bounding box (class, leftX, topY, rightX, bottomY) were converted into YOLO format with bounding box (class, x-center, y-center, width, height). The training was done with a batch size of 16 for 20000 epochs with the model architecture built with Darknet53 as the backbone, Adam as the optimizer, and initializing the remaining parameters as per [7]. Training took around 26 hours to complete.

YOLOv4: It was trained with similar parameters, i.e. training data fed into a model with a batch size of 16 for 20000 epochs as of YOLOv3 with bounding box annotations same as the latter. The model used a modified version of Darknet53 known as CSPDarknet53 as its backbone, keeping the optimizer the same as Adam. Other parameters have been set as per [8] resulting in a training time similar to YOLOv3 i.e. 26 hours

SSD: For training our custom dataset on SSD, the text file annotations with bounding box (class, leftX, topY, rightX, bottomY) were converted to xml files for Pascal VOC with bounding box (xmin-top left, ymin-top left, xmax-bottom right, ymax-bottom right). Using VGG-16 as the backbone and SGD as the optimizer, this model was trained with a batch size of 8 for 120 epochs with 1000 steps per epoch at decreasing learning rate. The rest of the parameters were according to [9]. The total training time taken was 10 hours.

## 6.4.4 Software and Hardware

Google Colaboratory, a notebook environment supporting python execution through remote servers on Linux platforms, is worked upon for training and testing the custom dataset. Google Colab provides a remote platform with a memory of nearly 100GB out of which 33GB is available for the user, 2 Intel(R) Xeon(R) CPU @ 2.20GHz CPUs, RAM of 13GB, and a Tesla K80 GPU having 2496 CUDA cores and 12GB GDDR5 VRAM. The free version has a maximum allocation per user time limit of 12 hours per server with an idle cut-off time of 90 minutes to make server allocation more efficient.

## 6.4.5 Performance Parameters

The developed DL-based food detection models are evaluated against the most commonly employed evaluation metrics such as Accuracy, Precision, Recall, F1 score, IoU, and mAP. These metrics are defined as below:

*Accuracy*: It is a measure of the percentage of correctly predicted images. Mathematically, it is defined in equation 6.7.

|  |  |
| --- | --- |
|  | (6.7) |

*Precision*: Precision indicates the number of true positive predicted objects that actually belong to the true positive class in the dataset. Mathematically, it is represented by the ratio of the number of images that were predicted correctly to the number of images that were predicted to be belonging to a category irrespective of correctness (equation 6.8).

|  |  |
| --- | --- |
|  | (6.8) |

*Recall*: Recall signifies the fraction of the number of images that were predicted correctly to the total number of images that belonged to the category (equation 6.9).

|  |  |
| --- | --- |
|  | (6.9) |

*F1 Score*: The harmonic mean of precision and recall of the model is represented by the F1 Score. It is defined in equation 6.10.

|  |  |
| --- | --- |
|  | (6.10) |

*IoU*: It is a measure that evaluates the resemblance of the predicted bounding box with the ground truth bounding box. IoU of value 1 or 100% depicts that the predicted bounding box is completely overlapping the ground truth bounding box. It can be mathematically denoted in equation 6.11.

|  |  |
| --- | --- |
|  | (6.11) |

*mAP*: Mean average precision is the mean of AP values of all different classes as represented by equation 6.12.

|  |  |
| --- | --- |
|  | (6.12) |

where True Positive (TP) indicates the total number of images that belonged to the same category and were predicted in the same category. True Negative (TN) denotes the total number of images which were not belonging to the specific category and were not predicted in that category. False Positive (FP) signifies the total number of images that didn’t belong to a category but were predicted in the category. False Negative (FN) demonstrates the total number of images that belonged to a category but were not predicted in that category.

# 6.5 Results

YOLOv3 showed the best out of 3 results giving an mAP of 74.6%, average IoU (Intersection over Union) of 56.85%, the precision of 0.70, recall of 0.67, and an F1 score of 0.69. YOLOv4 was just slightly behind YOLOv3, YOLOv4 achieved an mAP of 71.3%, an average IoU of 52.78%, and an F1 score of 0.67. SSD being the weakest of the three mentioned models gave an mAP of 13.6% with no detection for Burrito and Hot Dog classes on the validation dataset. Relative to YOLOv4, YOLOv3 achieved a greater mAP by 4.61%. Results in Fig 6.5 and Table 6.3 show that YOLOv3 and YOLOv4 performed much better than SSD. YOLOv3 performed slightly better than YOLOv4 mainly due to YOLOv3's very high accuracy to classify Burrito, 25.6% more relative to YOLOv4. Both these models have been indifferent to the number of images in each class as YOLOv3 has classified burrito with the highest precision of 93.6% and largest class, fast food, with 58.9% precision that can be verified from Table 6.3.

The cause can be inferred from the intuition that YOLOv4 focuses on providing better accuracies on large sample sizes with better speed. Burrito having only 322 samples even after augmentation has a sample size towards the lower end of the size spectrum, thus, nullifying the edge of YOLOv4 over YOLOv3 in the class. SSD has performed poorly on weak classes like Burrito and Hot dog. It has not performed comparatively well on a strong dataset because it gives only 9.1% accuracy on the Fast Food dataset which is the strongest among the whole dataset as seen in Fig 6.5 and Table 6.3. It can happen due to confusion among similar object categories such as pasta and pizza because multiple categories share locations.

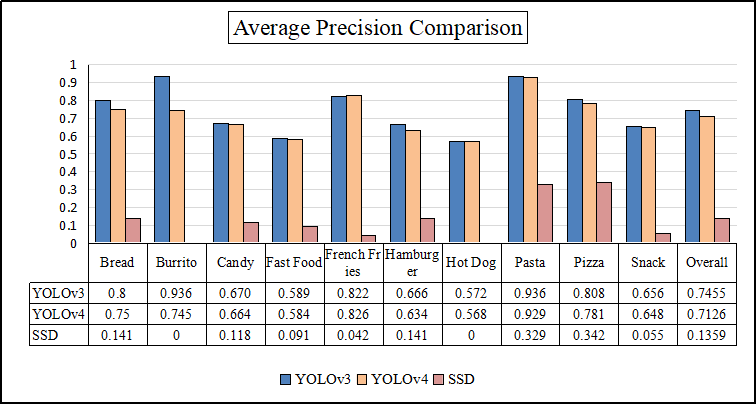


Fig 6.5***.*** Comparison of Average Precision

For the 10 classes Bread, Burrito, Candy, Fast Food, French Fries, Hamburger, Hot Dog, Pasta, Pizza, and Snack classes YOLOv3 achieved AP of 80.0%, 93.6%, 67%, 58.9%, 82.2%, 66.6%, 57.2%, 93.6%, 80.8%, and 65.6% respectively. YOLOv4 closely following behind resulted in 75%, 74.5%, 66.4%, 58.4%, 82.6%, 63.4%, 56.8%, 92.9%, 78.1%, and 64.8% of AP for the respective classes compared to 14.1%, 0%, 1.8%, 9.1%, 4.2%, 14.1%, 0%, 32.9%, 34.2%, and 5.5% of AP for SSD. Further Table 6.4 displays the normalized confusion matrices obtained for each class based on the 3 experimented models YOLOv3, YOLOv4, and SSD. Test results for YOLOv3, YOLOv4, and SSD on one of the test images are shown in Fig 6.6. For the tested image of Pizza, YOLOv3 predicted correctly with an accuracy of 94%, YOLOv4 followed behind with 86% accuracy. In contrast, SSD gave correct prediction with least accuracy of 37%.

Table 6.3 Detection Results of Models on the Augmented Dataset

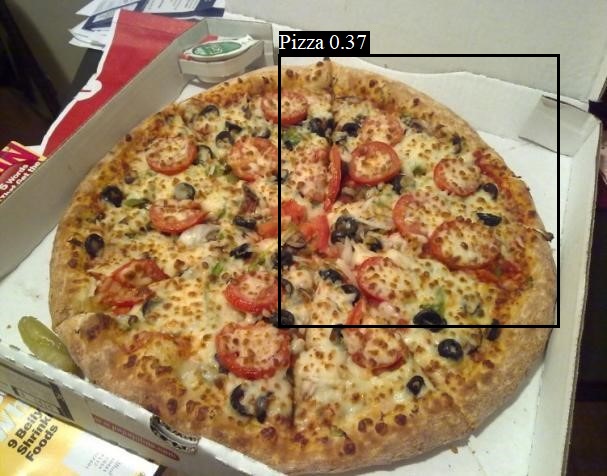
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Class** | **Total Images** | **Model** | **Precision** | **Recall** | **Average Precision(AP)** | **F1 Score** |
| Bread | 3556 | YOLOv3 | 0.73 | 0.78 | 0.800 | 0.75 |
| YOLOv4 | 0.66 | 0.72 | 0.75 | 0.69 |
| SSD | 0.28 | 0.19 | 0.14 | 0.23 |
| Burrito | **322** | YOLOv3 | 0.89 | 0.55 | 0.93 | 0.68 |
| YOLOv4 | 0.66 | 0.25 | 0.74 | 0.36 |
| SSD | 0 | 0 | **0** | 0 |
| Candy | 1470 | YOLOv3 | 0.87 | 0.6 | 0.67 | 0.71 |
| YOLOv4 | 0.81 | 0.61 | 0.66 | 0.7 |
| SSD | 0.26 | 0.1 | 0.12 | 0.14 |
| Fast Food | **7448** | YOLOv3 | 0.61 | 0.72 | 0.59 | 0.66 |
| YOLOv4 | 0.55 | 0.85 | 0.58 | 0.67 |
| SSD | 0.16 | 0.06 | 0.09 | 0.09 |
| French Fries | 2312 | YOLOv3 | 0.76 | 0.55 | 0.82 | 0.64 |
| YOLOv4 | 0.76 | 0.52 | 0.82 | 0.62 |
| SSD | 0.16 | 0.66 | 0.04 | 0.26 |
| Hamburger | 2604 | YOLOv3 | 0.58 | 0.43 | 0.66 | 0.49 |
| YOLOv4 | 0.58 | 0.39 | 0.63 | 0.47 |
| SSD | 0.28 | 0.19 | 0.14 | 0.23 |
| Hot Dog | 686 | YOLOv3 | 0.77 | 0.31 | 0.57 | 0.44 |
| YOLOv4 | 0.71 | 0.3 | 0.56 | 0.42 |
| SSD | 0 | 0 | 0 | 0 |
| Pasta | 1876 | YOLOv3 | 0.93 | 0.78 | 0.93 | 0.85 |
| YOLOv4 | 0.91 | 0.7 | 0.93 | 0.79 |
| SSD | 0.37 | 0.46 | 0.33 | 0.41 |
| Pizza | 2828 | YOLOv3 | 0.88 | 0.63 | 0.81 | 0.73 |
| YOLOv4 | 0.83 | 0.6 | 0.78 | 0.7 |
| SSD | 0.42 | 0.42 | 0.34 | 0.42 |
| Snack | 5250 | YOLOv3 | 0.72 | 0.73 | 0.65 | 0.72 |
| YOLOv4 | 0.66 | 0.78 | 0.65 | 0.72 |
| SSD | 0.07 | 0.5 | 0.05 | 0.12 |
| **Overall** | **28352** | YOLOv3 | **0.7** | **0.67** | **0.74** | **0.68** |
| YOLOv4 | **0.64** | **0.69** | **0.71** | **0.66** |
| SSD | **0.26** | **0.21** | **0.13** | **0.23** |

Table 6.4 Resultant Normalized Confusion Matrices

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** | **Model** | **Actual** | **Predicted:  Negative** | **Predicted: Positive** |
| Hamburger | YOLOv3 | Actual: Others | 0.49 | 0.12 |
| Actual: Hamburger | 0.22 | 0.17 |
| YOLOv4 | Actual: Others | 0.44 | 0.12 |
| Actual: Hamburger | 0.27 | 0.17 |
| SSD | Actual: Others | 0.01 | 0.32 |
| Actual: Hamburger | 0.54 | 0.13 |
| Pizza | YOLOv3 | Actual: Others | 0.53 | 0.04 |
| Actual: Pizza | 0.16 | 0.28 |
| YOLOv4 | Actual: Others | 0.48 | 0.06 |
| Actual: Pizza | 0.18 | 0.28 |
| SSD | Actual: Others | 0.1 | 0.33 |
| Actual: Pizza | 0.33 | 0.24 |
| Pasta | YOLOv3 | Actual: Others | **0.72** | 0.02 |
| Actual: Pasta | 0.06 | 0.21 |
| YOLOv4 | Actual: Others | 0.69 | 0.02 |
| Actual: Pasta | 0.09 | 0.21 |
| SSD | Actual: Others | 0.24 | 0.4 |
| Actual: Pasta | 0.27 | 0.23 |
| Candy | YOLOv3 | Actual: Others | 0.24 | 0.06 |
| Actual: Candy | 0.28 | 0.41 |
| YOLOv4 | Actual: Others | 0.2 | 0.1 |
| Actual: Candy | 0.27 | 0.43 |
| SSD | Actual: Others | 0.04 | 0.21 |
| Actual: Candy | 0.67 | 0.07 |
| Burrito | YOLOv3 | Actual: Others | 0.75 | 0.02 |
| Actual: Burrito | 0.11 | 0.13 |
| YOLOv4 | Actual: Others | 0.45 | 0.06 |
| Actual: Burrito | 0.36 | 0.12 |
| SSD | Actual: Others | 0 | 0 |
| Actual: Burrito | 0 | 0 |
| Snack | YOLOv3 | Actual: Others | 0.19 | 0.18 |
| Actual: Snack | 0.17 | 0.46 |
| YOLOv4 | Actual: Others | 0.16 | 0.24 |
| Actual: Snack | 0.13 | **0.47** |
| SSD | Actual: Others | 0.04 | 0.27 |
| Actual: Snack | 0.02 | 0.02 |
| Bread | YOLOv3 | Actual: Others | 0.48 | 0.12 |
| Actual: Bread | 0.09 | 0.31 |
| YOLOv4 | Actual: Others | 0.42 | 0.16 |
| Actual: Bread | 0.12 | 0.3 |
| SSD | Actual: Others | 0.06 | 0.27 |
| Actual: Bread | 0.49 | 0.18 |
| French Fries | YOLOv3 | Actual: Others | 0.65 | 0.05 |
| Actual: French Fries | 0.13 | 0.16 |
| YOLOv4 | Actual: Others | 0.63 | 0.05 |
| Actual: French Fries | 0.15 | 0.16 |
| SSD | Actual: Others | 0.46 | 0.42 |
| Actual: French Fries | 0.04 | 0.08 |
| Hot Dog | YOLOv3 | Actual: Others | 0.36 | 0.05 |
| Actual: Hot Dog | 0.4 | 0.18 |
| YOLOv4 | Actual: Others | 0.29 | 0.08 |
| Actual: Hot Dog | 0.44 | 0.19 |
| SSD | Actual: Others | 0 | 0 |
| Actual: Hot Dog | 0 | 0 |
| Fast Food | YOLOv3 | Actual: Others | 0.18 | 0.26 |
| Actual: Fast Food | 0.16 | 0.4 |
| YOLOv4 | Actual: Others | 0.29 | 0.08 |
| Actual: Fast Food | 0.44 | 0.19 |
| SSD | Actual: Others | 0.14 | 0.35 |
| Actual: Fast Food | 0.07 | 0.43 |



1. (b)

(c) (d)

Fig 6.6. (a) Test Image (b) Prediction for YOLOv3 (c) Prediction for YOLOv4 (d) Prediction for SSD

## 6.6 Conclusion and future scope

The research resulted in the successful training of three models for food detection on ten different food classes that can be readily applied in various applications. YOLOv3, achieved mAP as high as 93.6% on Burrito as well as Pasta giving the best results, and is suitable for use as a common food detector for local vendors as well as food chains. Real-time inference speed was achieved for all three models. Application-based on these detectors to measure nutritional levels can be developed to avoid malnutrition. YOLOv3 and YOLOv4 have been able to detect all images even though they failed at classifying some of them correctly. A simple multiple item counter application can be created using these two models to overcome the drawbacks of conventional sensor-based counters used in industries. Moreover, food detection becomes an essential daily utility for people requiring visual aids. This can be realized by combining trained detection techniques with NLP modules. Furthermore, these results will help future researchers working on similar projects giving them a few intuitions.

**Author Contributions:** Conceptualization, H.G., and O.P.V.; Methodology, B.S., M.K., and D.; Software, B.S., M.K., and D.; Formal analysis, B.S., M.K., and D.; Validation, H.G., and O.P.V.; Investigation, B.S., M.K., and D..; Data curation, B.S., M.K., and D.; Writing-original draft preparation, B.S., M.K., and D.; Visualization, H.G., and O.P.V.; Writing review and editing, H.G., and O.P.V.; Supervision, H.G., and O.P.V. All authors have read and agreed to the published version of the manuscript.

**Bibliography**

[1] World Health Organisation (2012) Comprehensive implementation plan on maternal, infant, and young child nutrition https://www.who.int/news-room/fact-sheets/detail/malnutrition

[2] Earman, A. M. (2016). Eye Safety for Proximity Sensing Using Infrared Light-emitting Diodes. *Renesas*.

[3] Kumar, S., Yadav, D., Gupta, H., Verma, O. P., Ansari, I. A., & Ahn, C. W. (2021). A Novel YOLOv3 Algorithm-Based Deep Learning Approach for Waste Segregation: Towards Smart Waste Management. *Electronics*, 10, 1. https://doi.org/10.3390/electronics10010014

[4] Song, H., Liang, H., Li, H., Dai, Z., & Yun, X. (2019). Vision-based vehicle detection and counting system using deep learning in highway scenes. *European Transport Research Review*, 11(1). https://doi.org/10.1186/s12544-019-0390-4

[5] Gupta, H., Varshney, H., Sharma, T. K., Pachauri, N., & Verma, O. P. (2021). Comparative performance analysis of quantum machine learning with deep learning for diabetes prediction. *Complex & Intelligent Systems*. https://doi.org/110.1007/s40747-021-00398-7

[6] Lawal, O. M. (2021). YOLOMuskmelon: Quest for Fruit Detection Speed and Accuracy Using Deep Learning. *IEEE Access*, 9, 15221-15227. 10.1109/ACCESS.2021.3053167

[7] Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *arXiv*, 6.

[8] Bochkovskiy, A., Wang, C. Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv*, 17.

[9] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. *arXiv*

[10] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *arXiv*, 10.

[11] Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The PASCAL Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision, 88(2), 303–338.*

[12] Mutreja, G., Aggarwal, A., Thakur, R., Tiwari, S. S., & Deshpande, S. (2020). Comparative Assessment of Different Deep Learning Models for Aircraft Detection. 2020 *International Conference for Emerging Technology (INCET)*, 6.O15

[13] Khan, A. M. (2021). Vehicle and Pedestrian Detection using YOLOv3 and YOLOv4 for Self Driving Cars. *California State University San Marcos*, 30.

[14] Gupta H, Verma OP (2021) Monitoring and surveillance of urban road traffic using low altitude drone images: a deep learning approach. *Multimedia Tools and Applications* 1–21. https://doi.org/10.1007/s11042-021-11146-x

[15] Zhao, Z.-Q., Zheng, P., Xu, S.-T., & Wu, X. (2019). Object Detection With Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11), 3212-3232. 10.1109/TNNLS.2018.2876865

[16] J. Redmon. Darknet: Open source neural networks in c. http://pjreddie.com/darknet/, 2013–2016

[17] Yun, S., Han, D., Oh, S. J., Chun, S., & Yoo, Y. (2019). CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features. *arXiv*. 1905.04899

[18] Zhaohui, Z., Wang, P., Liu, W., Li, J., Ye, R., & Ren, D. (2019). Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. *arXiv*. 1911.08287

[19] He, K., Zhang, X., Ren, S., & Sun, J. (2014). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *Springer International Publishing*, 346–361. 10.1007/978-3-319-10578-9\_23

[20] Lazebnik, S., Schmid, C., & Ponce, J. (2006). . Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In 2006 *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (CVPR'06) (Vol. 2, pp. 2169-2178). 10.1109/CVPR.2006.68

[21] Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *arXiv*. 1409.1556

[22] Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., Duerig, T., & Ferrari, V. (2020). The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*.

[23] Heras J (2018) CLoDSA, GitHub Repository.https://github.com/joheras/CLoDSA