SN Computer Science (2021) 2:441
https://doi.org/10.1007/s42979-021-00837-9

ORIGINAL RESEARCH

Maturity Detection of Tomatoes Using Deep Learning

Sahil Amol Mutha[1] · Akshat Mayur Shah[1] · Mohammed Zakee Ahmed[1]

Received: 21 November 2020 / Accepted: 24 August 2021 / Published online: 3 September 2021

Abstract
Agriculture 5.0 primarily constitutes the use of artificial intelligence and robotics as a hybrid technology that can automate a major portion of agriculture. Artificial intelligence will provide a cognitive skill to a computer to detect diseases that may occur in various eatables, such as fruits and vegetables, that can lead to a potential loss of crop. Also, the maturity that is the ripening status of these fruits and vegetables can be estimated to decide harvesting time. There are numerous ways to estimate the ripening status based on size, shape, texture, or color. Most of these features can be captured with images or video and decision-making is made possible by applying deep learning and artificial intelligence. After the decision-making stage, the fruit or vegetables can be plucked with a robotic arm. In this paper, we demonstrate the use of deep learning to detect the maturity specific of tomatoes. We create a customized dataset of images and use convolution neural networks along with the popular object detection model, YOLO v3 to detect the maturity of the tomatoes and pinpoint their location.

Keywords Deep learning · CNN · YOLO v3

Introduction

Artificial Intelligence is one of the most hiking and superior technologies nowadays which has limitless applications. Some of the major technologies in agriculture using artificial intelligence are autonomous tractors, agricultural robotics, controlling pest infestations, soil and crops health monitoring, AI-based drones on fields, precision farming with predictive analytics and many more [1].

UN report has predicted that the world might face a food supply crisis, due to a mismatch between population growth and fruits and vegetable growth [1]. So, we need to increase the production of fruits and vegetables. To achieve this, we need to increase the speed of farming, i.e., need to do fast farming. Fast farming with precision can be achieved with the help of cutting-edge technology, one of which is artificial intelligence paired with robotics.

Particular features such as object detection or classification of different fruits or vegetables can be done using machine learning/deep learning–neural networks, but the accuracy and losses vary for different models and its algorithm. Several deep learning architectures are available for varied applications such as artificial neural networks (ANN), convolution neural networks (CNN), recurrent neural networks (RNN), Modular neural network (MNN), region-based convolutional networks (R-CNN), radial basis function neural network, and YOLO (You only look Once), whereas machine learning is divided into supervised machine learning, unsupervised machine learning, reinforcement learning and more. But in general, deep learning–neural networks work better for images and videos, i.e., for 2D and 3D data, whereas machine learning works better for particular collected data, i.e., mathematical data or numeral data. We have used tomatoes as a primary object to investigate their maturity level and the machine learning model of tomatoes is prepared for detecting its maturity in three levels. These three levels are classified as:

Level-1: Unripped (growing stage)—0–50%
Level-2: Ripped (eatable/usable stage)—51–100%

This article is part of the topical collection "Intelligent Computing and Networking" guest edited by Sangeeta Vhatkar, Seyedali Mirjalili, Jeril Kuriakose, P.D. Nemade, Arvind W. Kiwelekare, Ashok Sharma and Godson Dsilva.

✉ Akshat Mayur Shah
akshatshahm@gmail.com

Sahil Amol Mutha
sahilmutha1999@gmail.com

Mohammed Zakee Ahmed
zakeeahmed@pict.edu

[1] Department of E&TC, Pune Institute of Computer Technology, Pune, India

Level-3: Damaged.

These levels of maturity will help farmers to decide the right harvesting time in automatic mode and also this may assist the robotic arm to pluck the right fruit or vegetable. This may be a part of precision agriculture which may improve decision-making for harvesting time.

Implementation

We have used CNN and YOLO over our created customized datasets for training and testing them.

Using CNN

Initially, we used CNN, which is a class of neural networks that are based on shared-weights architecture and translation invariance characteristics. CNN image classifications take an input image, process it and classify it under certain categories.

Figure 1 shows the architecture of the CNN model, which trains and tests each input image by passing it through a series of convolution layers with filters (Kernals), Pooling, fully connected (FC) layers, and apply SoftMax function to classify an object with probabilistic values between 0 and 1 [2].

The CNN Model

We built our customized dataset consisting of 4370 images of which 3064 images we used for training and 1306 images for validation followed by which we have split this dataset into training (70%) and testing (30%). We referred to and extracted the “Fruits 360” dataset which consisted of 131 different classes of fruits and vegetables followed by which we also included some real-world images of various tomatoes under different conditions.

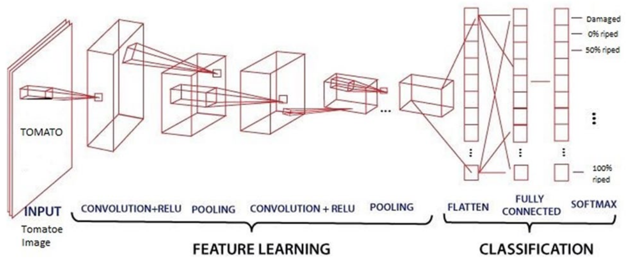


Fig. 1 A typical CNN architecture

Flowchart of the CNN Model

Figure 2 shows the flowchart of our CNN model. This flowchart is generated to help visualize what is going on inside our model while training images and thereby analyze its workflow simply and abstractly.

Here, we used different featuring layers such as MaxPooling 2D, Convolution 2D, and ReLU, followed by flattening layers and FC layers, and finally applied SoftMax activation function over it. We also used some dropout layers (data augmentation) to counter the issue of overfitting.

Training of CNN Model

Figure 3 shows how we trained the dataset for 20 epochs, where we observed a drastic reduction in loss from 0.6484 in the first epoch to 0.0767 in the last (20th) epoch, whereas an exponential increase in accuracy from 74.06% till 97.40%.

Testing of CNN Model

Figure 4 shows testing of our CNN model over the dataset for 20 epochs, where we observed a reduction in loss from 0.7027 in the initial (1st) epoch till 0.0927 in the last (20th) epoch, whereas an increase in accuracy from 70.20% till 96.80%.

Using YOLOv3 (You Only Look Once Version3)

CNN object detection algorithm uses regions to localize the object within the image, but in YOLO a single convolutional network predicts the bounding boxes and accordingly assigns class probabilities for these boxes as shown in Fig. 5. In YOLO, we take an image and split it into an S × S grid. We take m bounding boxes within each of the grid. The network provides class probability and offsets values, for each of the bounding boxes. Bounding boxes having the class probability greater than a threshold value are selected which is further used to locate the object within the image. Compared to other object detection algorithms, YOLO is faster as it is of faster magnitude (45 frames per second), which differentiates it completely from other models [3].

Bounding Box Prediction and Cost Function Calculation

The network predicts 4 coordinates for each bounding box, t_x, t_y, t_w, t_h. If the bounding box prior has width and height p_w, p_h and the cell is offset from the top left corner of the image by (c_x, c_y) and the, sigmoid function $\sigma()$ then the predictions correspond to:

$$b_x = \sigma(t_x) + c_x b_y = \sigma(t_y) + c_y b_w = p_w e^{t_w} b_h = p_h e^{t_h}.$$

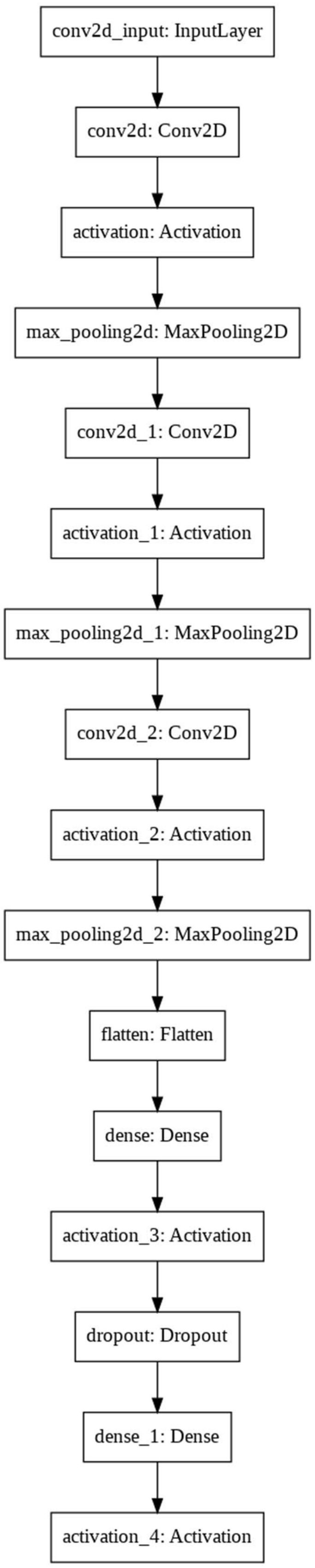
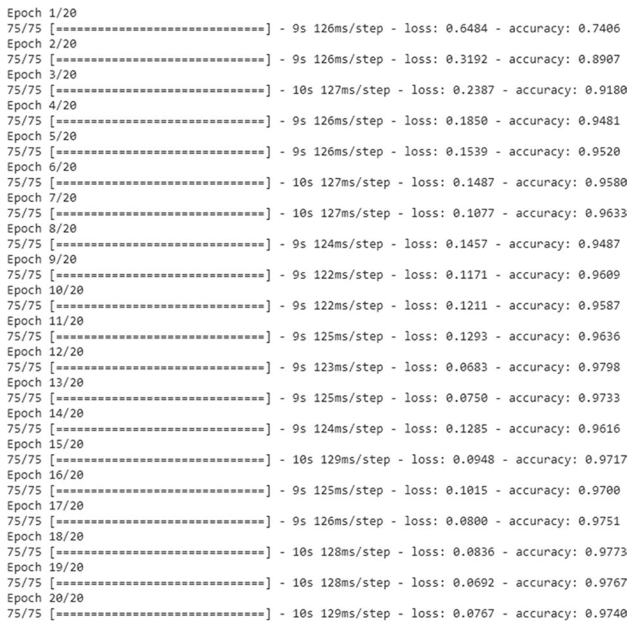
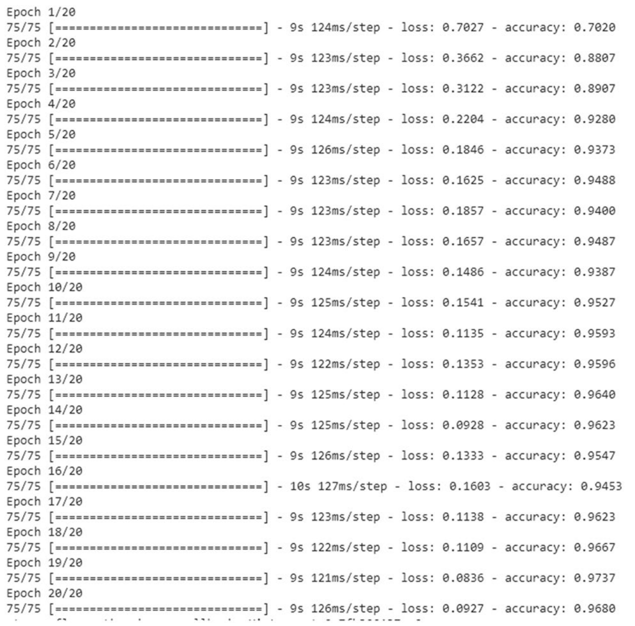


Fig. 2 Flowchart of CNN model for the customized dataset



```
Epoch 1/20
75/75 [==============================] - 9s 126ms/step - loss: 0.6484 - accuracy: 0.7406
Epoch 2/20
75/75 [==============================] - 9s 126ms/step - loss: 0.3192 - accuracy: 0.8907
Epoch 3/20
75/75 [==============================] - 10s 127ms/step - loss: 0.2387 - accuracy: 0.9180
Epoch 4/20
75/75 [==============================] - 9s 126ms/step - loss: 0.1850 - accuracy: 0.9481
Epoch 5/20
75/75 [==============================] - 9s 126ms/step - loss: 0.1539 - accuracy: 0.9520
Epoch 6/20
75/75 [==============================] - 10s 127ms/step - loss: 0.1487 - accuracy: 0.9580
Epoch 7/20
75/75 [==============================] - 10s 127ms/step - loss: 0.1077 - accuracy: 0.9633
Epoch 8/20
75/75 [==============================] - 9s 124ms/step - loss: 0.1457 - accuracy: 0.9487
Epoch 9/20
75/75 [==============================] - 9s 122ms/step - loss: 0.1171 - accuracy: 0.9609
Epoch 10/20
75/75 [==============================] - 9s 122ms/step - loss: 0.1211 - accuracy: 0.9587
Epoch 11/20
75/75 [==============================] - 9s 125ms/step - loss: 0.1293 - accuracy: 0.9636
Epoch 12/20
75/75 [==============================] - 9s 123ms/step - loss: 0.0683 - accuracy: 0.9798
Epoch 13/20
75/75 [==============================] - 9s 125ms/step - loss: 0.0750 - accuracy: 0.9733
Epoch 14/20
75/75 [==============================] - 9s 124ms/step - loss: 0.1285 - accuracy: 0.9616
Epoch 15/20
75/75 [==============================] - 10s 129ms/step - loss: 0.0948 - accuracy: 0.9717
Epoch 16/20
75/75 [==============================] - 9s 125ms/step - loss: 0.1015 - accuracy: 0.9700
Epoch 17/20
75/75 [==============================] - 9s 126ms/step - loss: 0.0800 - accuracy: 0.9751
Epoch 18/20
75/75 [==============================] - 10s 128ms/step - loss: 0.0836 - accuracy: 0.9773
Epoch 19/20
75/75 [==============================] - 10s 128ms/step - loss: 0.0692 - accuracy: 0.9767
Epoch 20/20
75/75 [==============================] - 10s 129ms/step - loss: 0.0767 - accuracy: 0.9740
```

Fig. 3 Training of CNN model



```
Epoch 1/20
75/75 [==============================] - 9s 124ms/step - loss: 0.7027 - accuracy: 0.7020
Epoch 2/20
75/75 [==============================] - 9s 123ms/step - loss: 0.3662 - accuracy: 0.8807
Epoch 3/20
75/75 [==============================] - 9s 123ms/step - loss: 0.3122 - accuracy: 0.8907
Epoch 4/20
75/75 [==============================] - 9s 124ms/step - loss: 0.2204 - accuracy: 0.9280
Epoch 5/20
75/75 [==============================] - 9s 126ms/step - loss: 0.1846 - accuracy: 0.9373
Epoch 6/20
75/75 [==============================] - 9s 123ms/step - loss: 0.1625 - accuracy: 0.9488
Epoch 7/20
75/75 [==============================] - 9s 123ms/step - loss: 0.1857 - accuracy: 0.9400
Epoch 8/20
75/75 [==============================] - 9s 123ms/step - loss: 0.1657 - accuracy: 0.9487
Epoch 9/20
75/75 [==============================] - 9s 124ms/step - loss: 0.1486 - accuracy: 0.9387
Epoch 10/20
75/75 [==============================] - 9s 125ms/step - loss: 0.1541 - accuracy: 0.9527
Epoch 11/20
75/75 [==============================] - 9s 124ms/step - loss: 0.1135 - accuracy: 0.9593
Epoch 12/20
75/75 [==============================] - 9s 122ms/step - loss: 0.1353 - accuracy: 0.9596
Epoch 13/20
75/75 [==============================] - 9s 125ms/step - loss: 0.1128 - accuracy: 0.9640
Epoch 14/20
75/75 [==============================] - 9s 125ms/step - loss: 0.0928 - accuracy: 0.9623
Epoch 15/20
75/75 [==============================] - 9s 126ms/step - loss: 0.1333 - accuracy: 0.9547
Epoch 16/20
75/75 [==============================] - 10s 127ms/step - loss: 0.1603 - accuracy: 0.9453
Epoch 17/20
75/75 [==============================] - 9s 123ms/step - loss: 0.1138 - accuracy: 0.9623
Epoch 18/20
75/75 [==============================] - 9s 122ms/step - loss: 0.1109 - accuracy: 0.9667
Epoch 19/20
75/75 [==============================] - 9s 121ms/step - loss: 0.0836 - accuracy: 0.9737
Epoch 20/20
75/75 [==============================] - 9s 126ms/step - loss: 0.0927 - accuracy: 0.9680
```

Fig. 4 Testing of CNN model

YOLOv3 uses logistics regression for predicting the score for each object detected and places a bounding box. If the bounding box prior overlaps a ground truth object more than other bounding boxes, the corresponding detected score should be 1. The default predefined threshold for bounding boxes is 0.5. For other bounding boxes prior with overlap greater the default threshold, they incur no cost. One boundary box is assigned to each ground truth object prior only. If

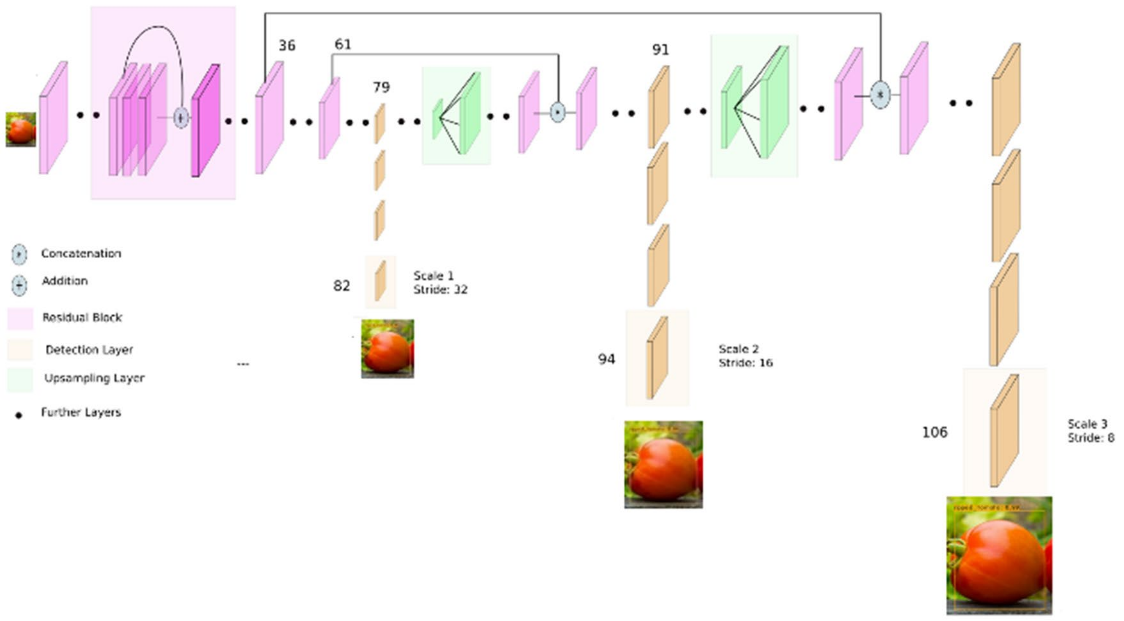


Fig. 5 Yolo v3 architecture

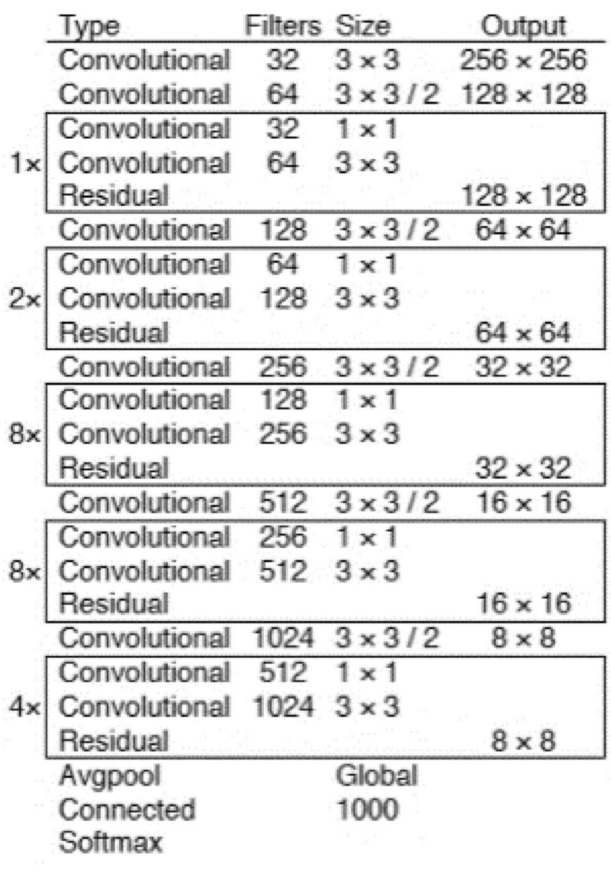
a bounding box prior is not assigned, it incurs no classification and localization lost. It just incurs confidence loss on objectness. t_x and t_y are used to calculate the loss [3].

Feature Extractor Network (Darknet-53)

The network used in YOLOv3 is a hybrid approach between the network used in YOLOv2(Darknet-19) and the residual network, so it has added advantages of both coming up with some shortcut connections. It has 53 convolutional layers as shown in Fig. 6, so they call it Darknet-53. Darknet-53 mainly composes of 3 × 3 and 1 × 1 filters with skip connections as the residual network in ResNet [3]. In computing, the measure of computer performance mainly in the fields of scientific computations requiring floating-point calculations, i.e., more accurate measures that are called FLOPS (Floating Point Operations Per Second). ResNet-152 has more BFLOP (billion floating-point operations) than Darknet-53, still, Darknet-53 achieves the same classification accuracy that to 2 × faster.

Custom Object Dataset

We built a dataset consisting of 3500 images of three classes namely ripped, unripe and damaged tomato. We gave annotations to each image as per the YOLO model required (in. txt format). We made a configuration file and customized it according to our model requirements setting max-batches to 6000 (i.e., 2000 × number_of_classes).



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig. 6 Network of Darknet-53 [3]

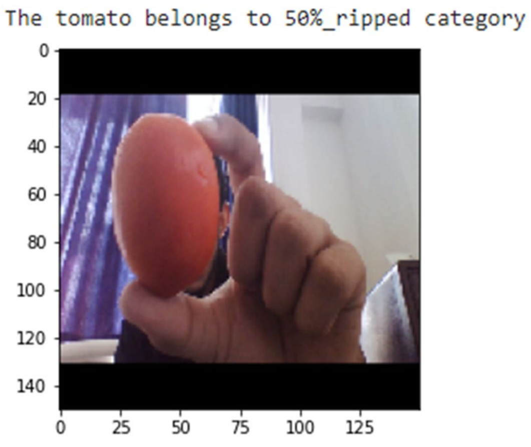


Fig. 7 Result-1 of CNN

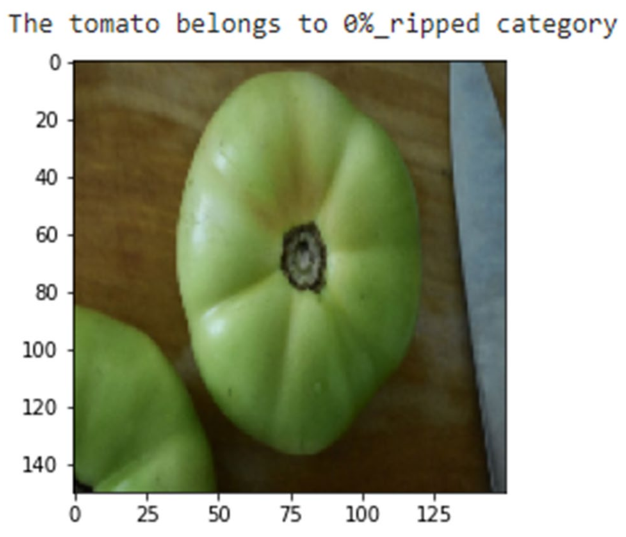


Fig. 8 Result-2 of CNN

Results

Results Using CNN

We tested our model using the OpenCV library over some images captured from our laptop camera and achieved the result of one of the images as shown in Fig. 7 and we even tested some images taken randomly from various web sources, one of whose result is shown in Fig. 8.

In Fig. 9, the upper graph represents training and testing accuracy vs epoch which signifies that the model has overfitted as training accuracy is higher than testing accuracy. The lower graph of Fig. 9 represents training and testing loss vs epoch.

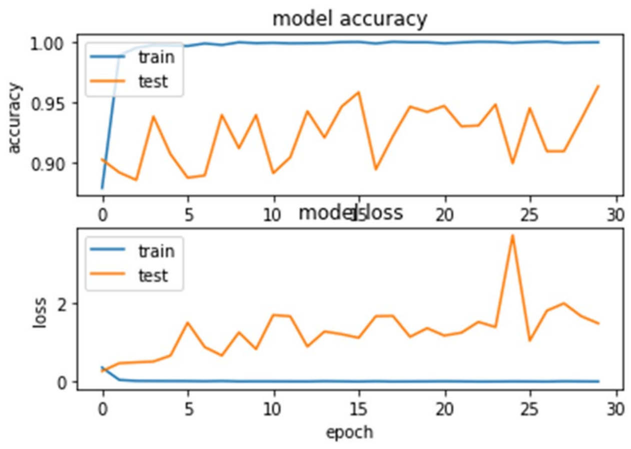


Fig. 9 Training and testing accuracy vs epoch, training and testing loss vs epoch

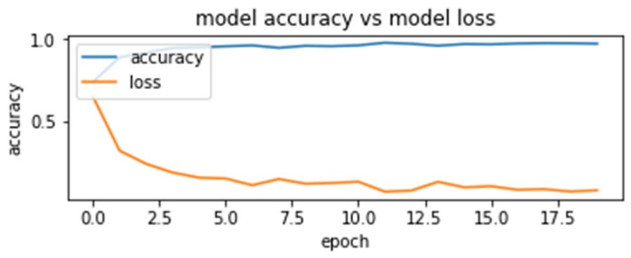


Fig. 10 Accuracy vs loss graph of our CNN model

Figure 10 shows the accuracy vs loss graph of our CNN model. We can observe from this graph that the accuracy of our model is better and consistently increasing, whereas loss decreasing as the number of epochs increases.

From Fig. 3, we can see that our CNN model is taking 9 s/10 s on each epoch for training which is very time-consuming, and thus it is hard to implement it in real-time applications. CNN object detection algorithm uses regions to localize object within the image, so to counter these problems of CNN, we used YOLOv3 model for detecting maturity of tomatoes.

Results Using YOLOv3 (You Only Look Once version3)

We classified the ripening, unripe and damaging of tomato with great accuracy for ripening being 99.2%, unripe being 94.34% and for damaged being of 90.23% as shown in Figs. 11, 12, and 13. Thus, the YOLOv3 model performed better than the CNN model in terms of accuracy, loss, and mainly the time constraint, while considering the detection of tomatoes as individuals (single tomato) as well as in overlapped regions.

We have achieved exponential growth in our accuracy and have reduced loss from 4.87 in the first iteration of training

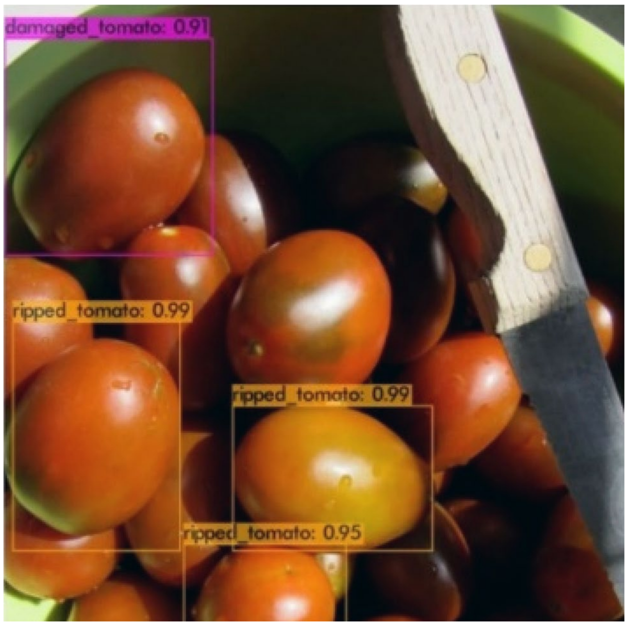


Fig. 11 Result-1 of YOLO (3)

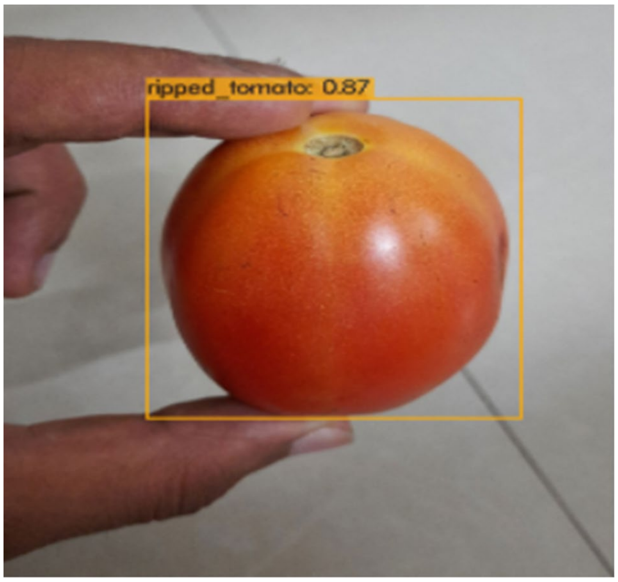


Fig. 12 Result-2 of YOLO (3)

to 0.039604 in the last (6000) iteration which can be referred from the resulted graph shown in Fig. 12.

Table 1 shows us the result comparison of both CNN and YOLO v3 model over same images, whereas Tables 2 and 3 depict the average confidence level and average loss occurred after prediction over both the models.

By observing Tables 2 and 3, we can see that the average loss and confidence level/accuracy of YOLO v3 is much better than CNN.

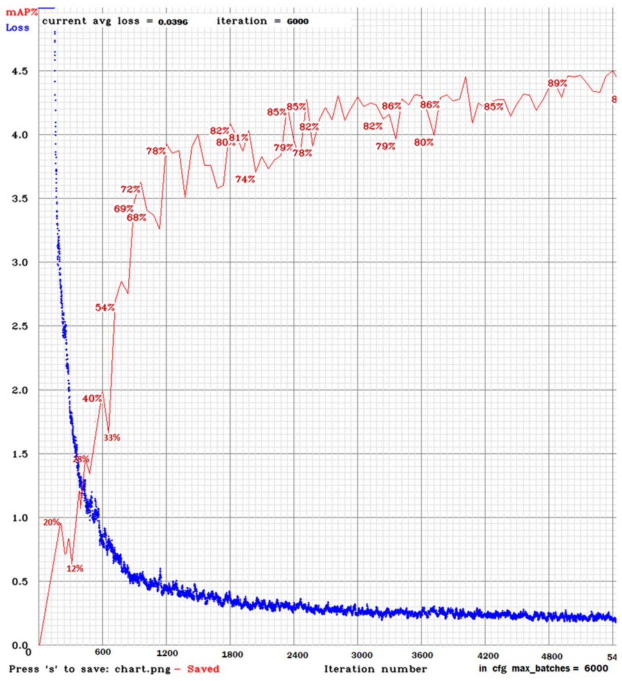


Fig. 13 Accuracy while training the model of Yolov3

Table 1 Result comparison of both models

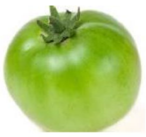
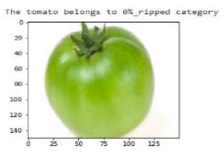
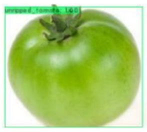
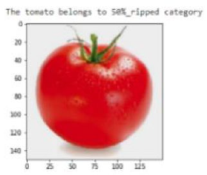
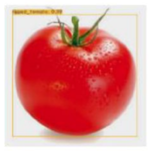
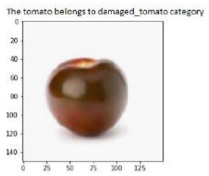
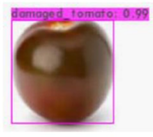
Original Image	CNN Output	Yolo v3 Output
		
		
		

Table 2 Comparison of decision-making and average confidence level

Actual category	No of images tested on	CNN—output		Yolo v3—output	
		No of right decisions	Avg confi-dence level (%)	No of right decisions	Avg. confi-dence level (%)
Unripen	8	6 Unripen	91	8 Unripen	95
Ripen	12	9 Ripen	98	11 Ripen	98
Damaged	3	1 Damaged	83	2 Damaged	91

Table 3 Loss Comparison of both techniques

Model	Loss
CNN	0.6484
Yolo v3:	0.039604

Conclusion

The present work confirms the fact that detecting the maturity of tomatoes at different levels can be achieved using deep learning models CNN and YOLO with an average accuracy of 90.67% and 94.67%, respectively.

The proposed work is based on the maturity level of the tomatoes. Initially, we used CNN model for training the dataset. So, we made our custom dataset using some of the images from the fruits 360 datasets and by adding some of our images. We trained this custom model which consisted of 3 different classes for tomatoes and got a good training accuracy and testing accuracy of 97.40% and 96.90%, respectively, as shown in Figs. 3 and 4, respectively. By doing this, we proved that our model is overfitted over this dataset and also got to know that CNN consumes very high time which reached approximately 9 s per epoch, which catered us to switch to the YOLO model. Yolo v3 has a rate of more than 40 FPS along with which it provides faster execution and also has very subtle complexity which concluded us for it is the best-fitted model in all the measures. So, from all the research and analysis, we get to know that the Yolo v3 can be also used for detecting the maturity of various fruits or vegetables as we did the same for tomatoes and can be further deployed into agriculture application.

Declarations

Conflict of Interest All authors declare that they have no conflict of interest.

References

1. Murugesan R, Sudarsanam SK, Malathi G, Vijayakumar V, Neelanarayanan V, Venugopal R, Rekha D, Saha S, Bajaj R, Miral A, Malolan V. Artificial intelligence and agriculture 5.0. Int J Recent Technol Eng. 2019;8(2):2277–3878.
2. Russakovsky O, Deng J, Su H, et al. ImageNet large scale visual recognition challenge. Int J Comput Vision. 2015;115(3):211–52.
3. Redmon J, Farhadi A. YOLOv3: an incremental improvement. Washington: University of Washington; 2018.