Deep Learning For Fruits Image Recognition[1]

N. Gheza

21st June 2019

Abstract

A subsection of object classification and detection from images is fruit recognition. This work presents the training of different Deep Neural Networks (DNN) for fruit images classification and detection. At first, a Deep Convolutional Neural Network (DCNN) is trained from scratch on two benchmark datasets for fruit classification. Then, the same datasets are used to fine-tune two pre-trained models, Inception v3 and MobileNetV2, to improve the performance of the fruit classifier. At last, using a fruit benchmark dataset with bounding boxes annotations, fruit detection is achieved by fine-tuning a Faster R-CNN architecture using ResNet-101 pre-trained model as base architecture.

1 Introduction

Being able to develop an accurate fruit detection system can significantly contribute to many applications such as fully automated harvesting, fruit grading, etc. This work presents the training of different architectures for image classification and detection. Three benchmark datasets are used for the training and evaluation of the models. The first two datasets are used to train multiple models using different Deep Convolutional Neural Network (DCNN) architectures for image classification. The third dataset, instead, is used to train a Faster R-CNN model for image detection.

The remainder of this paper consists of the following. Section 2 introduces the state-of-the-art for fruits recognition from images. Section 3 presents the benchmarking datasets used for the training and testing the multiple architectures presented in the paper and a comparison between them. In Section 4, the SoA fruit classification techniques are introduced. In particular, a Convolutional Neural Network (DCNN) trained from scratch and the use of transfer learning to fine-tune pre-trained models. Fruit detection which involves spatial localization, in addition to classification, is presented in Section 5 with an introduction to the Faster R-CNN architecture and the training of multiple models for the task of finding the position and classifying multiple fruits in an image. Results of Fruit classification and detection are discussed in Section 6. Conclusions are finally drawn in Section 7.

2 State Of The Art

Many attempts have been made to use neural networks and deep learning for fruits recognition. This section will review several of those attempts. The computer vision task dealing with classification and detection of objects of a certain class (e.g., 'cat', 'dog', etc) in images made substantial progress during the last 5 years. A reason for the strong interest and considerable advances in this area is the arrival of Deep Convolutional Neural Networks (DCNN) [1].

Hussain et al. [7] propose a fruit recognition algorithm based on Deep Convolutional Neural Network (DCNN) for 15 classes of fruit images and a total of 44406 images. Another fruit recognition system based on Deep Convolutional Neural Network (DCNN) is proposed by [9] in 2018. The authors introduced a new high-quality dataset of images of different fruits. The dataset, which was named Fruits-360, contains 103 different fruits and a total of 71022 images. The same paper also presents the results of different numerical experiments for training a Deep Convolutional Neural Network (DCNN) to recognise fruits. The previous research used the big amount of images to train the models from scratch. This process requires a lot of time,

[1] This thesis was prepared in partial fulfillment of the requirements for the Degree of Bachelor of Science in Data Science and Knowledge Engineering, Maastricht University, Supervisors: Alexia Briassouli, Gerasimos Spanakis.

data and computational power. Femling et al. [2] describe an approach that avoids the cost of training from scratch in a system that can identify fruits and vegetables in the retail market using images captured by a video camera. For the fruits and vegetables classification they used different Deep Convolutional Neural Network (DCNN) pre-trained architectures and applied transfer learning by fine-tuning these pre-trained architectures to their dataset.

In the previous applications the authors aimed at building image classification system for the specific purpose of classifying pictures of fruits. An even more challenging problem when dealing with objects recognition is finding its position in the image. A novel approach for detecting fruits and their location in images using Deep Neural Network (DNN) is presented in [12]. In this paper, the authors present a rapid training and real-time fruit detection system based on Faster Region-based Convolutional Neural Networks (Faster R-CNN) that can be adapted to various types of fruits with a minimum number of training images.

3 Datasets

Benchmarking datasets have become available for the training and testing of various Deep Neural Networks (DNNs) and their objective comparison. Table 1 introduces the datasets used in this paper by showing the number of images and categories for each dataset. Following, each dataset will be presented and described.

Dataset	# images	# categories
Dataset 1 [9]	71022	103
Dataset 2 [7]	44406	15
Dataset 3 [12]	614	7

Table 1: Number of images and categories in each dataset.

3.1 Dataset 1

The first dataset - Fruit-360 - created by H. Murean and M. Oltean [9], contains 61934 images of fruits spread across 90 labels. The images were obtained by filming the fruits while being rotated by a motor and then extracting frames. Successively, a dedicated algorithm extracted the fruit from the background and scaled the images down to 100x100 pixels. An example image is shown in Figure 1.

Figure 1: Example image from Dataset 1 - Fruit-360.

3.2 Dataset 2

The second dataset, created by Hussain et al. [7], contains 44406 fruit images sorted in 15 categories of fruits. The pictures were captured using an HD Logitech web camera with a resolution of 320x258x3 pixels. An example image is shown in Figure 2.

Figure 2: Example image from Dataset 2.

3.3 Dataset 3

The last dataset, created by Sa et al. [12], is a very small dataset compared to the previous two. In fact, this dataset only contains 563 fruit images sorted in 7 different categories of fruits. The special characteristics of this dataset, compared to the others, is that it contains annotations with the location of each fruit in the images together with its classification. An example image is shown in Figure 3.

4 Fruit Classification

Multiple techniques are available when one wants to develop an image classifier. In this

Figure 3: Example image from Dataset 3 - with drawn bounding boxes based on annotations.

particular case, the goal is to be able to classify multiple categories of fruits from an image with one or multiple fruits. Two datasets, Dataset 1 and 2, will be used in the training of multiple image classifiers. These two datasets are quite different: the first dataset [9] contains very simple images with just the picture of the fruit extracted from the background, as in Figure 1. The second one [7], instead, contains pictures of fruits on a tray and the images are more noisy. Figure 2 is an example of such images.

4.1 Deep C-NN

In this paper, the first implementation of a fruit image classifier was done by training a Deep Convolutional Neural Network (DCNN) with the same architecture as in [9]. This architecture is given in Table 2.

Layer type	Dimensions	Output
Convolutional	5 x 5 x 4	16
Max pooling	2 x 2 stride 2	-
Convolutional	5 x 5 x 16	32
Max pooling	2 x 2 stride 2	-
Convolutional	5 x 5 x 4	64
Max pooling	2 x 2 stride 2	-
Convolutional	5 x 5 x 16	128
Max pooling	2 x 2 stride 2	-
Fully connected	5 x 5 x 128	1024
Fully connected	1024	256
Softmax	256	Num. classes

Table 2: Architecture of the neural network used in this paper.

The first layer is a convolutional layer which applies 16 5x5 filters to the input image. This layer that retains the most important features is then followed by a max pooling layer with a filter of shape 2x2 with stride 2. Max pooling is a sample-based discretization process that down-sample an input representation by applying a max filter to non-overlapping subregions of the initial representation. The stride of 2 is used to avoid overlaps between the regions represented by the filter [8]. The second convolutional layer applies 32 5x5 filters to the output of the first two layers and similarly to the first convolutional layer it applies a max pooling layer of shape 2x2 with stride 2. The third convolutional layer applies 64 5x5 filters followed by another max pooling layer of shape 2x2 with stride 2. The fourth, and final, convolutional layer applies 128 5x5 filters also followed by a max pooling layer. Following the convolutional and max pooling layers there is the first fully connected layer with 5x5x128 inputs and 1024 outputs. The following layer is another fully connected layer with 1024 inputs and 256 outputs. Finally, the last layer is a softmax loss layer with 256 inputs.

The softmax layer take an N-dimensional vector of real numbers and transform it into a vector of real number in range (0, 1) that adds up to 1. The resulting vector contains as many numbers as the number of classes (based on each dataset) which represent the probability of belonging to each specific class (or fruit category).

This vector is then used to compute the Cross Entropy loss during training. The Cross Entropy loss indicates the distance between what the model believes an image is and what the image really is. The Cross-Entropy Loss is defined as Figure 1. Where t_i and s_i are the groundtruth and the CNN score for each class $_i$ in C and n is the number of classes (plus the background).

$$CE = -\sum_{i=1}^{C'=n} t_i log(s_i) \tag{1}$$

4.1.1 Training from scratch

The Deep Convolutional Neural Network (DCNN) architecture previously explained was used to build a baseline model for both Dataset 1 and Dataset 2. For both the datasets, the network was trained over 75000 epochs with a batch size of 60 images taken from the training set. Every 100 steps, accuracy is computed using cross-validation. Finally, the test set is used to compute the final accuracy.

The initial learning rate for the Deep Convolutional Neural Network (DCNN) is 0.001 and it is updated every 100 epochs using Eq. 2 until

it reaches the final learning rate 0.00001.

$$\eta = max(\eta - \alpha * \eta * 0.9, H) \quad (2)$$

Where: η is the learning rate, α is the accuracy and H is the final learning rate.

The training dataset is augmented by preprocessing the RGB images. This takes place by applying random hue and saturation changes, horizontal and vertical random flips and converting them to the HSV colorspace and to graycale and merging them.

As one could expect, the results of the Deep Convolutional Neural Network (DCNN) trained on Dataset 1 and Dataset 2 are not the same. The results for each dataset are presented in the results section.

4.1.2 Results - Dataset 1

The Deep Convolutional Neural Network (DCNN) trained from scratch on the Fruit-360 dataset with 100x100 pixels images obtained good results scoring a final test accuracy of 96.3%.

The number of incorrectly classified fruit images is 648 on a test set of 17845 images. Thus, only about 3.7% of the images were misclassified. Figure 4 shows some example images used in the evaluation of the model trained on the Fruit-360 dataset. The first two images on the top represent an orange and a mandarin which were correctly classified from the network. In the bottom the other two images are a lemon and a pear. The lemon image was classified by the model as a pear while the pear was classified as a cherry. This could be because both colors and shape of lemons and pears are quite similar. The reason for the pear being misclassified as a cherry is less clear and to fully investigate this many more experiments would be required.

4.1.3 Results - Dataset 2

The same Deep Convolutional Neural Network (DCNN) on Dataset 2 with 150x150 pixels images was able to attain a final testing accuracy of 98.7%. On a test set of 8888 fruit images the number of misclassified images is 114 that is about 1.3% of the total test set.

Figure 5 shows four examples of testing images used during the evaluation of the model. The model correctly classified the first two images showing that it does not learn only to detect colours but also shapes and other features as the apples in the images have different colours. The third and fourth image show a banana and a tomato. The model misclassified the banana for a kiwi and the tomato with an apple. The reason for the misclassification could be that the images have a lot of lights but more experiments would be needed to fully understand why it is happening.

Figure 4: Example of testing images used in the evaluation of the model. The top two images shows an orange and a mandarin. The bottom images show a lemon and a pear.

Figure 5: Example of testing images used in the evaluation of the model. The top two images shows two different type of apples with different colours. The bottom images show a banana and a tomato.

4.2 Transfer Learning

Training an image classifier from scratch requires a lot of labeled data, computing power and especially time. With transfer learning, it is possible to reduce the cost of training by taking a pre-trained model and reusing it to train a new classifier. The classifier uses feature extrac-

tion capabilities from SoA classifiers and train a new classification layer on top of it [10].

4.2.1 Training via Transfer Learning

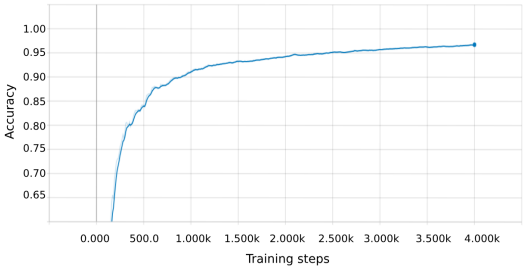
As for the Deep Convolutional Neural Network (DCNN), both dataset Dataset 1 and Dataset 2 were used to train two separate models. Transfer learning requires a pre-trained model which will be fine-tuned to the type of fruit in an image. Different pre-trained model architectures are publicly available, but comparing all architectures to each other is not the aim of this work. Instead, as was done by Femlinget al. [2], the Inception and MobileNet architectures were taken into consideration for the training of multiple models for the two datasets.

Inception v3 is an opensource image recognition model that achieved more than 78.1% on the ImageNet dataset [14]. Mobile and embedded vision applications require lightweight architectures. MobileNet uses depth-wise separable convolutions to build light weight Deep Neural Networks (DNN) [5]. In 2018, Google published MobileNetV2. MobileNetV2 is a significant improvement to its predecessor, MobileNetV1, and a push to the state of the art in mobile image recognition [13]. Both models were trained over 4000 training steps with the data split in 70% for training, 20% for validation, 10% for testing. Learning rate is set to 0.01. Results for the Inception v3 and MobileNetV2 models are presented for each dataset in the following results sections.

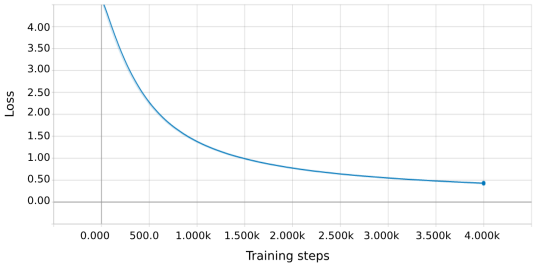
4.2.2 Results - Dataset 1

The fine-tuned Inception v3 architecture obtained good results on Dataset 1 scoring a test accuracy of 97.2% and cross-entropy loss of 0.4289. From Figure 6 it can be seen the test accuracy and cross-entropy loss over 4000 steps while fine-tuning the Inception v3 pre-trained model. Figure 6a shows that the accuracy rapidly increases in the first 1000 training steps and keeps increasing during the next 3000 steps. Figure 6b shows a similar behaviour for the cross-entropy loss, but as one would expect the value is decreasing instead of increasing as for the accuracy. Looking into the mismatches, 200 of the 7161 test fruit images were misclassified with another fruit.

MobileNetV2 was able to attain a test accuracy of 99.7% with cross-entropy loss of 0.1243



(a)



(b)

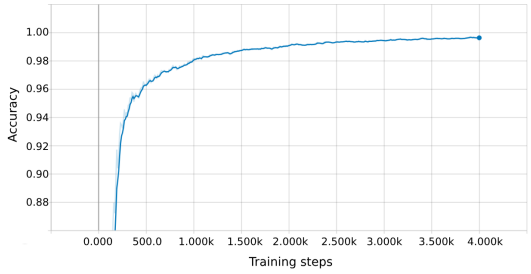
Figure 6: (a) Test accuracy over 4000 steps for Dataset 1 with Inception, (b) Test cross-entropy loss over 4000 steps for Dataset 1 with Inception.

for the Fruit-360 dataset.

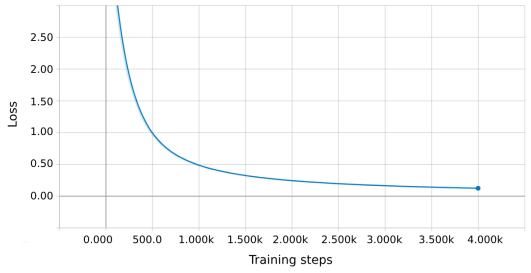
Figure 7a and 7b shows the test accuracy and cross-entropy loss over 4000 steps while fine-tuning MobileNetV2. From Figure 7a it can be seen that the accuracy rapidly improves in the first 1000 iterations and then it slowly improves in the next 3000 iterations. Looking at Figure 7b it can be seen that the cross-entropy loss quickly decreases in the first 1000 steps and then slowly decreases in the next 3000 steps. The number of misclassified fruit images is only 22 of the 7161 images used for testing. Thus MobileNetV2 obtained an almost 10-fold improvement compared to the Inception v3 model.

4.2.3 Results - Dataset 2

The Inception v3 model scored 95% testing accuracy with cross-entropy loss 0.2317. Figure 8 shows test accuracy and cross-entropy loss over 4000 iterations. From Figure 8a it can be seen that the testing accuracy quickly increase in the first 500 iterations and then keeps increasing slowly until the end of training. Similarly, Figure 8b shows that the loss follow the same trend

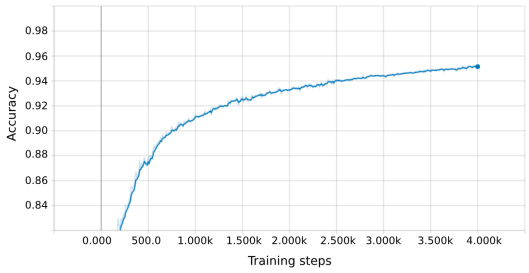


(a)

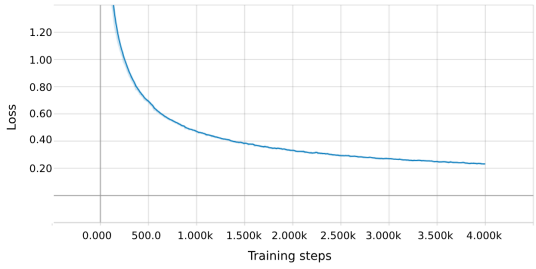


(b)

Figure 7: (a) Test accuracy over 4000 steps for Dataset 1 with MobileNet, (b) Test cross-entropy loss over 4000 steps for Dataset 1 with MobileNet.



(a)



(b)

Figure 8: (a) Test accuracy over 4000 steps for Dataset 2 with Inception, (b) Test cross-entropy loss over 4000 steps for Dataset 2 with Inception.

as the accuracy and quickly decreases in the first 500 iterations to then slowly keep decreasing for 4000 iterations. On a test set of 4409 fruit images the Inception v3 model misclassified 237 images.

For the Dataset 2, MobileNetV2 obtained an accuracy of 98.5% with cross-entropy loss 0.08116. Figure 9 shows test accuracy and cross-entropy over 4000 steps. Both accuracy and loss keep increasing until the end of training after 4000 iterations. Similarly as for the model trained on Dataset 1, both accuracy and cross-entropy loss quickly increase/decrease until around 1000 iterations and keep improving for the next 3000 steps. The fine-tuned MobileNetV2 mode misclassified 65 images on a test dataset of 4409 where the Inception v3 model had misclassified 237.

Dataset	Scratch	InceptionV3	MobileNetV2
Dataset 1	96.3%	97.2%	99.7%
Dataset 2	98.7%	95%	98.5%

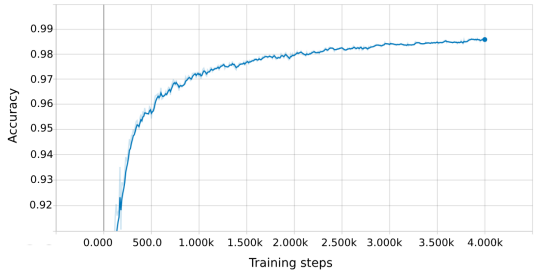
Table 3: Results for Fruit Classification

5 Fruit Detection

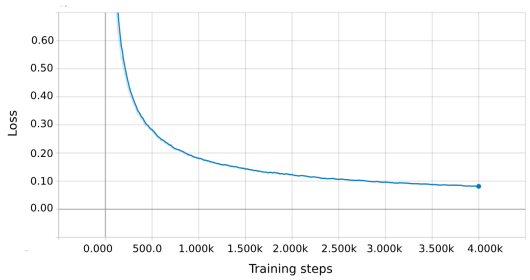
To be able to build an accurate fruit detection (i.e. spatial localization and recognition) system is a key element for fruit yield estimation and automated harvesting [12]. When training a fruit detection system, annotations with fruit position and class together with the actual fruit images are needed. Finding a dataset with this information is not as easy as finding just images of specific classes. Sa et al. [12] published a dataset, Dataset 3, which contains bounding box annotations for multiple fruit images as one can see in Figure 3. Table 4 shows the total amount of images per fruit in Dataset 3 and the total number of images used for training and testing.

5.1 Faster R-CNN

The implementation of a fruit detection system was done by training a Faster R-CNN because as shown in [12] it can be trained on a small amount of training data. Faster R-CNN [11] is state-of-the-art object detection system with an



(a)



(b)

Figure 9: (a) Test accuracy over 4000 steps for Dataset 2 with MobileNet, (b) Test cross-entropy loss over 4000 steps for Dataset 2 with MobileNet.

Fruit	Training (# images)	Test (# images)	Total
Apple	51	13	64
Avocado	43	11	54
Pepper	98	22	120
Mango	136	34	170
Orange	45	12	57
Rockmelon	49	7	56
Strawberry	33	9	42

Table 4: Number of training and testing images for each fruit in Dataset 3.

architecture composed of two modules. The first module is a Deep Convolutional Neural Network (DCNN) that proposes regions, and the second module is the Fast R-CNN detector [3] that uses the proposed regions. The entire system is a single, unified network for object detection that uses a Region Proposal Network (RPN) [11] to find regions (bounding boxes) which may contain objects and a Fast Region-based Convolutional Neural Network (Fast R-CNN) that classify the content of each bounding box. The two networks share the same convolutional layers which can be a pre-trained Convolutional Neural Network (CNN) such as VGGNet or ResNet.

Figure 10 from [11] shows a high-level representation of the Faster R-CNN architecture. At first, the image goes through the convolutional layers and feature maps are extracted. A sliding window is then used in the Region Proposal Network (RPN) for each location over the feature map. For each location, k anchor boxes are used to generate region proposals which the classifier layer of the RPN uses to compute 2k scores that estimate probability of whether there is object or not for each proposal. The regression layer of the RPN has 4k outputs encoding the coordinates for each proposal. The k anchor boxes, called anchors, are centered at the sliding window from above and are associated with a scale and aspect ratio. The RPN uses 3 scales and 3 aspects ratios which yields to k = 9 anchors at each sliding position. With a WxH convolutional feature map, there are WHk anchors in total.

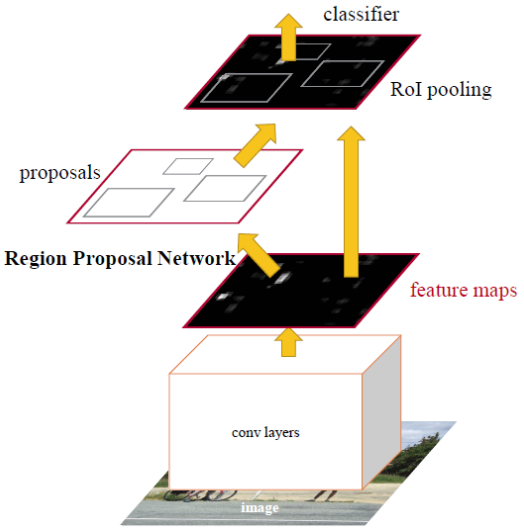


Figure 10: Faster R-CNN architecture.

The RPN loss function can be seen in Equation 3. The first term is the classification loss over 2 classes (the object is present or not). The second term is the regression loss of bounding boxes in the case an object is present, thus $(p_i, p_i^*) = 1$.

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i L_{reg}(t_i, t_i^*) \quad (3)$$

The RPN network is used to pre-check which anchor contains object. The anchors labelled

as positive are then passed to the detection network for detecting the object class and to return the bounding box of that object.

The Detection network is the Fast R-CNN [3]. It performs RoI pooling and then, the pooled area goes through a Convolutional Neural Network with two Fully Connected branches for class softmax and bounding box regressor that show class and location. Figure 11 from [3] shows the Fast R-CNN architecture.

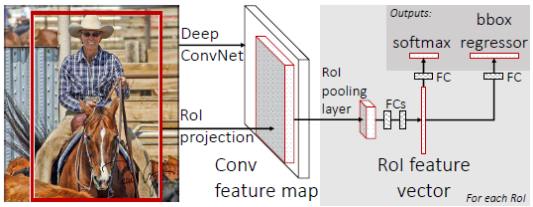


Figure 11: Fast R-CNN architecture.

5.2 Training via Transfer Learning

To train a Faster R-CNN model from scratch would be very costly effective in terms of data and GPU power. To train a Faster R-CNN model with a small amount of data as for the case of Dataset 3, fine-tuning was performed. Fine-tuning consists of adapting a pre-trained model to the new data [10]. In the case of the Faster R-CNN transfer learning is performed for both the RPN and Fast-RCNN network. The Faster-RCNN in this paper uses the ResNet-101 architecture to perform fine-tuning, as this scored the best performance.

ResNet-101 is a SoA architecture for image classification that, thanks to its very deep representations, was able to obtain a 28% relative improvement on the Faster R-CNN model trained on the COCO object detection dataset [4].

In this paper the network is trained in a one versus rest manner for each fruit category. This means that there are only two classes (e.g., 'orange', 'background') for each trained model. This is acceptable in real world applications because each fruit is cultivated separately due to economic reasons such as fertilisation, irrigation and the prevention of harmful diseases and insects [12]. The training of each model took 2 to 4 hours and around 200 to 400 epochs. In the next section, the results of training Faster R-CNN models for each fruit are presented. Moreover a model trained was trained on the entire dataset to detects and classify all the fruits using with single model. The model, which will be referred to as 'TuttiFrutti' is finally presented for experimenting purposes.

5.3 Results

A popular metric in measuring the accuracy of object detectors like Faster R-CNN is the Mean Average Precision (mAP). It computes how well a detector works across all classes. Moreover, it can be calculated across different IoU (Intersection over Union) thresholds. IoU measures the overlap between the ground truth and the predicted boundary [6].

In this work, models are evaluated using 0.5 as IoU threshold (notation mA@0.5). Average Recall with 0.50:0.95 IoU threshold is also taken into consideration to get more insights on the performance of each model.

Fruit	mAP@0.5	AR@0.50:0.95
Apple	89%	70%
Avocado	80%	62%
Pepper	54%	29%
Mango	93%	62%
Orange	88%	62%
Rockmelon	82%	53%
Strawberry	93%	64%
TuttiFrutti	80%	60%

Table 5: Mean Average Precision and Average Recall for each Faster R-CNN model.

Table 5 contains results for each model trained on a single fruit and the results for the 'TuttiFrutti' model trained on the whole dataset.

With the exception of the model traned to detect peppers, which obtained a mAP@0.5 of 54%, every other model was able to obtain a mAP@0.5 greater or equal to 80%. The models which obtained the highest mAP@0.5 are the strawberry and mango models which obtained a score of 93%. Figure 12c shows an instance of strawberry detection where fully grown strawberries and not yet ripe strawberries are both detected by the model. Figure 12h shows an example instance of mango detection. The model trained to detect apple obtained a mAP@0.5 of 89% and AR@0.50:0.95 of 70%. Figure 12a and 12b show two instances of apple detection with

different colors of apples. The model is capable to detect both kind of apples correctly. The orange detector model also scored good results with a mAP@0.5 of 88% and AR@0.50:0.95 of 62%. Figure 12d is interesting because it shows an instance of orange detection where one of the oranges is not in perfect conditions and still gets detected by the model. The rockmelon detector scored a mAP@0.5 of 82% with AR@0.50:0.95 of 53%. Figure 12f shows a perfect instance of rockmelon detection where one of the detected rockmelons has a finger on it. The model finedtuned for the detection of avocados was able to obtain a mAP@0.5 of 80% with AR@0.50:0.95 of 62%. Figure 12e shows an example instance of avocado detection.

The model trained on the entire dataset, 'TuttiFrutti', thus able to detect and classify 7 different fruits, obtained a mAP@0.5 of 80% and a AR@0.50:0.95 of 60%. An example instance of detecting multiple fruits in an image can be found in Figure 12i.

6 Discussion

The results from the fruit classification and fruit detection models presented in the previous sections are discussed in this section.

The first fruit classification model trained in this paper is the Deep Convolutional Neural Network (DCNN) trained from scratch. On Dataset 1 this model performed very similarly to the state-of-the-art for this dataset by Horea et al. [9]. The same architecture trained from scratch on Dataset 2 scored very good results with a low percentage of misclassified images. The second technique used for fruit classification is transfer learning. The Inception v3 and MobileNetV2 pre-trained architectures are fine tuned on both Dataset 1 and Dataset 2. Inception v3 obtained very good results on Dataset 1 increasing the model accuracy by 0.9%. The fine-tuning of Inception v3 on Dataset 2 instead did not performed as well as the first model trained from scratch. It is the fine-tuned MobileNetV2 model that scores the best results on both datasets. The fine-tuned model trained on Dataset 1 improved the accuracy by 3.4% compared to the model trained from scratch. Even though MobileNetV2 fine-tuned on Dataset 2 had a slightly lower accuracy than the model trained from scratch it improved the precision as the number of mismathces decreased.

In this paper the implementation of a fruit detection system was done by training a Faster R-CNN model using Dataset 3. For each of the 7 fruit classes present in Dataset 3 a Faster R-CNN model is fine-tuned using the ResNet-101 architecture. All models, with the exception of the pepper model, obtained good results for the selected metrics. The pepper model scored much lower accuracy and recall compared to the rest of models but still was able to detect and classify peppers in testing images.

7 Conclusions

In conclusion, this paper presented the two main state-of-the-art techniques for fruit images recognition: fruit images classification and fruit images detection.

In the developing of a fruit image classification system, a Deep Convolutional Neural Network (DCNN) trained from scratch and two other fine-tuned pre-trained architectures (InceptionV3, MobileNetV2) were compared to each other using two different datasets: the Fruit-360 dataset [9] and Dataset 2 [7].

To develop a fruit image detection system, multiple Faster R-CNN models were trained by fine-tuning the ResNet-101 architecture on Dataset 3 [12].

Finally, this work showed how Deep Learning techniques for both fruit image classification and fruit image detection are able to obtain high results comparable to SoA techniques for other image classification and detection areas.

References

[1] Shivang Agarwal, Jean Ogier Du Terrail, and Frédéric Jurie. Recent advances in object detection in the age of deep convolutional neural networks. *arXiv preprint arXiv:1809.03193*, 2018.

[2] Frida Femling, Adam Olsson, and Fernando Alonso-Fernandez. Fruit and vegetable identification using machine learning for retail applications. *arXiv preprint arXiv:1810.09811*, 2018.

[3] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[5] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[6] Jonathan Hui. map (mean average precision) for object detection, March 2018.

[7] Israr Hussain, Qianhua He, and Zhuliang Chen. Automatic fruit recognition based on dcnn for commercial source trace system.

[8] Fei-Fei Li, Andrej Karpathy, and Justin Johnson. Cs231n: Convolutional neural networks for visual recognition 2016. 2016.

[9] Horea Murean and Mihai Oltean. Fruit recognition from images using deep learning. *Acta Universitatis Sapientiae, Informatica*, 10:26–42, 06 2018.

[10] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

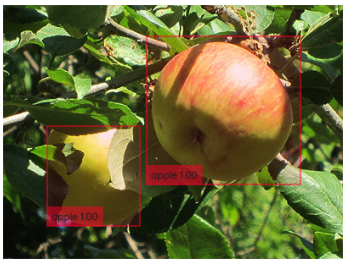
[11] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[12] Inkyu Sa, Zongyuan Ge, Feras Dayoub, Ben Upcroft, Tristan Perez, and Chris McCool. Deepfruits: A fruit detection system using deep neural networks. *Sensors*, 16(8):1222, 2016.

[13] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018.

[14] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

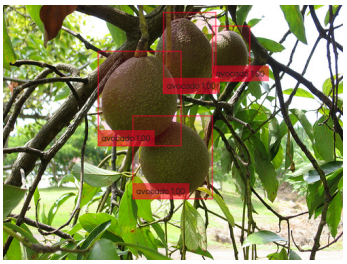
(a) Apple detection



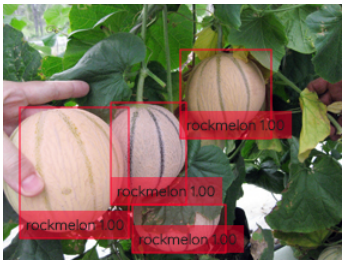
(b) Apple detection

(c) Strawberry detection

(d) Orange detection



(e) Avocado detection



(f) Rockmelon detection

(g) Mango detection

(h) Pepper detection

(i) Apple and Mango detection

Figure 12: Nine instances of detection results using test images from Dataset 3.