Everis Machine Learning Prize
Parts II and III Report

João Figueira and Tiago Almeida
(Dated: June 4, 2019)

I. INTRODUCTION

As students of Computer Science and Engineering at Instituto Superior Técnico that are partaking in the 2019 course on Machine Learning, a project sponsored by Everis was proposed to us in which we would have to develop or implement a neural network capable of recognizing fruits in images which may contain different fruits. This report accompanies our code delivery which can be found at: `https://github.com/joaoperfig/everismlprize`

II. CLASSIFICATION OF SINGLE FRUIT

The second part of the proposed challenge involves classifying images of a single fruit using machine learning computer vision methods. Our code for this part of the project can be found in the following notebook: `https://colab.research.google.com/drive/1t5ro-wostBrfCGlhVRK2OsYQM-mLWQJy`

A. Dataset

The dataset used for training in this section was the one developed in the previous stage of this project as our solution to the first part of the challenge. It includes over 10000 images of 41 different fruits for training and testing our network. The network is comprised of images scraped from google images, photos taken by ourselves, and the "Kaggle fruits 360" dataset[1]. In order to expedite the training and validation process we opted for only considering 7 classes of fruits (Apple, Lemon, Orange, Pear, Strawberry, Banana and Grapes) as these were the fruits that we saw as being the most common and we that we had plenty of training data.

This dataset was then divided into training, testing and validation subsets[2]. On the previous stage the dataset had already been divided into training (used for learning) and testing (independent dataset used to provide an unbiased evaluation of our final model). A validation set was needed to evaluate variations in the performance of our neural network, when we changed some of its properties (such as activation function used or the number of neurons in a certain layer of the network). We decided to use 25% of our whole dataset for validation therefore we took 25% of the training images and 25% of the testing images and used them for validation.

B. Neural Network

For the architecture of our model we opted for creating a convolutional neural network[3] (CNN) which is a deep learning model suited for image classification[4], we implemented this with python and using the Keras library over TensorFlow[5]. The input layer on our neural network converts the images into a three-dimensional array of [100x100x3] where the first 2 values are the length and width of images (all images are rescaled to 100x100) and the third is the number of colour channels. Our neural network has 2 2D convolutional layers with a ReLU activation function (returns 0 if the value is negative, otherwise the value itself)[6] a kernel size of [5x5] and increasing number of filters for low-level feature detection first[7]. The convolutional layers are followed by 2D pooling layers whose objective is to provide spatial variance, making the network able to recognize the fruits even when their appearance slightly varies[7] and finally, in order to prevent overfitting of our data, a 25% dropout rate was added. The output of the last convolutional layer is then flattened, and densely connected to 4 output nodes (our number of classes) in a regular (dense) layer with a sigmoid activation function[8]. Figure 1 shows the shape of the model.

We trained our neural network, using batch sizes of 16 images and a learning rate of 0.1 over 100 epochs.

The achieved accuracy when validating our network was 91%, but we found our neural network was underfitting our data, given that the validation loss was much lower than the training loss. To tackle this problem, we added more convolutional layers (with higher number of filters for high-level feature detection[7]) before flattening and more regular layers (with higher number of nodes) using a hyperbolic tangent activation function before our last layer. The obtained results after these changes had obvious improvements, with a validation loss value of 0.0456 and accuracy of 0.9877, much closer to the training loss value of 0.0487 and an accuracy of 0.9858, respectively.

C. Testing

After building the model and training it with the training and validation sets, we further evaluated it using the testing dataset for a final unbiased evaluation of our model. We achieved an accuracy of 98.72% and a loss of 4.49%. Figure 2 shows the graph displaying the evolution of the accuracy while testing our dataset as a function of

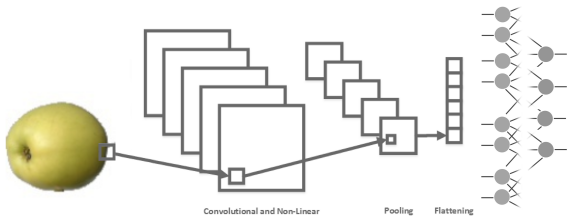

FIG. 1. Diagram showing the architecture of the used convolutional neural network.

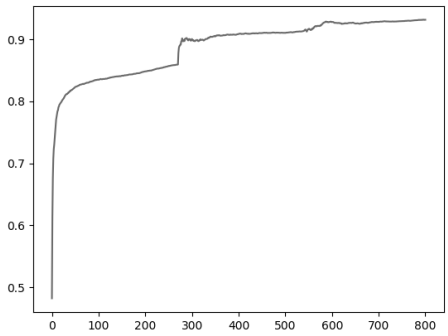

FIG. 2. Accuracy of model on Validation set over 800 batches of training.

the number of batches.

D. Result Analysis

Upon analysis of our experimental results we conclude that the usage of this type of neural network is very appropriate for image classification problems. Our network was capable of correctly classifying nearly every image from the dataset.

To further analyze the model's capabilities we used it to classify some other images of fruit that we obtained online. We found that there was a smaller accuracy when it came to these images. A bigger or more varied dataset might have yielded better results. Figures 3 and 4 are the classification results for two fruit images that were external to our dataset.

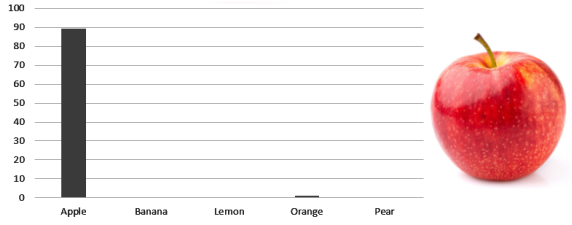

FIG. 3. Classification results for an image of an Apple.

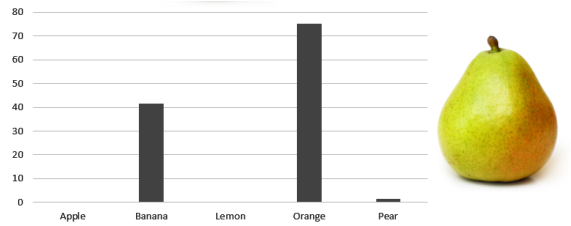

FIG. 4. Classification results for an image of a Pear. These are obviously inaccurate.

III. CLASSIFYING MULTIPLE FRUITS

The third part of the proposed challenge was detecting and classifying multiple (possibly different) fruits in a single image. For our implementation we analyzed a number of existing algorithms and found this part of the challenge to be far superior in complexity than the previous one. Most existing algorithms for multi-object detection are more complex in terms of implementation but also in terms of temporal and spacial complexity. The following notebook contains the working code for this part of the project: `https://colab.research.google.com/drive/1EMEY9F2PnnHiGUT7OoEjS1FzaipHtjvq`

A. Related Work

During our research we found multiple CNN algorithms for object detection and classification, most of them use window-sliding algorithms which are much slower when compared to state of the art algorithms like YOLO[9] and Single Shot MultiBox Detector (SSD)[10], which need only one forward pass through the network to predict object bounding boxes and class probabilities. Almost all of them make use of the Microsoft COCO library[11] that contains over 300.000 images but only 80 classes.

B. Approach Using YOLO

As mentioned in the Related Work, YOLO is a popular algorithm for object detection that is seen as state of the art for usage in these kinds of applications. As such, we chose it as our primary approach to tackle this problem.

1. *Dataset For Usage With YOLO*

To train a neural network for usage with YOLO, we required a Dataset of images with corresponding text annotation files. These files should list every object present in a picture along with their tags and pixel coordinates[12]. Given that we were trying to use our Dataset from the previous challenges, we attempted to convert our dataset

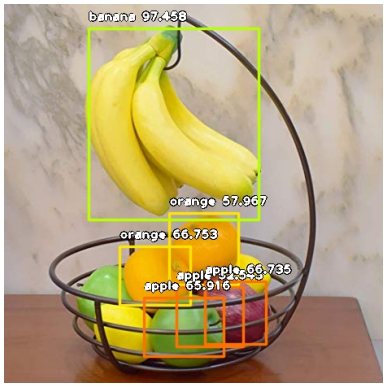

FIG. 5. Object detection results for an image of a basket of fruit, using a YOLO model trained with the COCO dataset.

of single fruit images to one with the desired format. To do this we implemented a simple script that would generate an annotation file for each image by assuming that each fruit completely filled the image. This would create the annotation files with the tag of the image as the tag of the single object and with the image dimensions as the coordinates of the object. In this stage we used Darkflow[13] to train and test our model.

For comparison, and since the COCO library has some labelled images of fruit, we also downloaded it for usage in this challenge. The library only has labels for apples, bananas, and oranges, which would normally be insufficient for a challenge like this. We do feel, however, that the much bigger and more accurately annotated dataset provided by COCO can provide better accuracy in the identification and classification of testing images. In this stage we used ImageAI[14] for model construction and testing.

2. *Result Analysis*

Using our dataset to train this neural network achieved sub-par results. The network could accurately classify most of the images from the training set but failed to classify any of the images from the testing dataset. We believe that this is due to the fact that we didn't have enough data, or that all of it was of a very specific nature. Our images are instances of fruit that are never partially obstructed and are never in a complex setting that might have rendered their detection more difficult. We also feel that to train a network with this kind of complexity we would need much more time with more powerful hardware.

When we used a model trained with the COCO library (which includes images with annotations of multiple objects), we achieved significantly better identification and classification results (as seen in Figures 5 and 6) the labelling is, however, constrained to the 3 aforementioned fruits (apples, bananas and oranges) as those are the only fruit classes covered by this dataset.

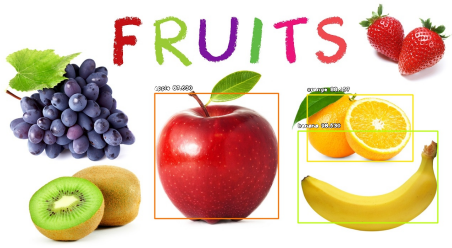

FIG. 6. Object detection results for an image of various fruits, using a YOLO model trained with the COCO dataset. The lack of certain fruit labels in the dataset is apparent here.

C. Our Approach

To try and build another solution to this challenge, from scratch, we decided to implement an algorithm that would allow us to use our model from the single-fruit challenge in a multi-fruit context. To do this we divided images into equally sized smaller pieces and ran those through our network. The model would then identify which had fruits and which didn't and mark the fruits on an exported output image.

1. *Classifying Non-Fruits*

The described algorithm requires that the classifier could work on images that had no fruits and correctly identify them as such. Upon testing our model on a few images of non-fruits we reached the conclusion that our model would identify them as some fruit most of the time. To fix this (that is probably a consequence of the lack of diversity in our dataset) we decided we needed to train our model to classify one additional class: everything but fruits. We added a set of completely random images that did not contain fruits to our dataset and retrained our neural network.

After retraining, our model was capable of correctly identifying around 97% of non-fruit images. Figure 7 shows an example of such a classification.

2. *Image Slicing and Certainty Maximization*

In order to align the sliced images with the fruits in the original image, we start by slicing into a small number of non-overlapping pieces. We incrementally slice into more pieces and all pieces are classified by our model. The final output is chosen as the sliced image that maximizes average certainty of the classifications of all its pieces. This algorithm needs to slice and classify the image multiple times therefore we expect it to be time consuming.

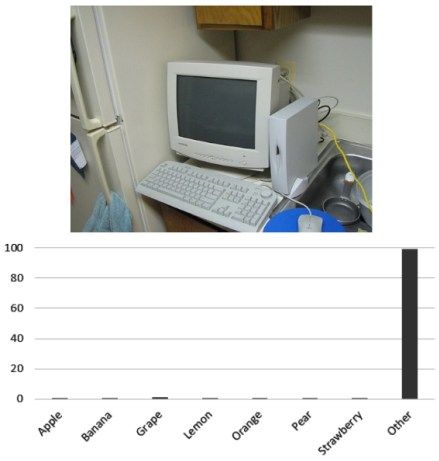

FIG. 7. Classification results for an image of a computer.

3. *Result Analysis*

After implementing, training, and testing our model and algorithm, we found that it could not always separate every single fruit into a different box due to the fact that all boxes are co-dependant. This could lead to some improper boxes with incorrect labels. As expected this algorithm was slower than the others we tested due to its almost brute-force nature. The problems of our model in the second challenge are also present here and some fruits could sometimes be incorrectly classified.

We do feel, however, that given the small Database and the lack of training of our model, this is still a very acceptable result that yielded some very accurate labelings. Figures 8 and 9 show examples of results and serve as way of comparing our method and dataset with the YOLO implementation using the COCO library. While our method does have some poor bounding boxes and the occasional mislabeling, it does provide the ability to classify a broader list of fruits.

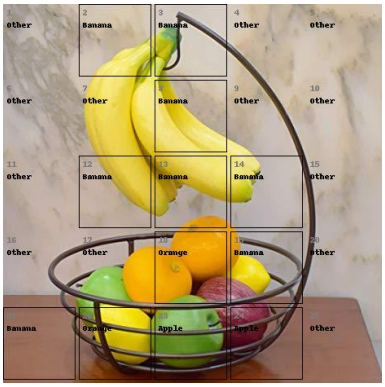

FIG. 8. Object detection results for an image of a basket of fruit, using our method and dataset. The drawbacks of the inaccurate model and the improper image cutting method are apparent here.

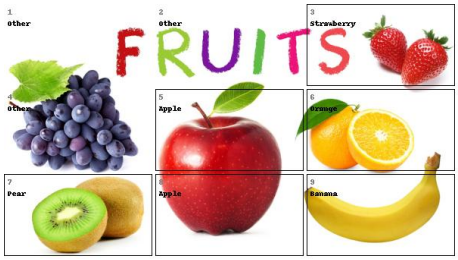

FIG. 9. Object detection results for an image of various fruits, using our method and dataset.

[1] H. M. M. Oltean, Fruits 360 dataset: A dataset of images containing fruits (2017).
[2] W. Sarle, What are the population, sample, training set, design set, validation set, and test set? (2002).
[3] I. S. A. Krizhevskyand G. Hinton, Imagenet classification with deep convolutional neural networks, (2011).
[4] K. S. D. Jaswal, Image classification using convolutional neural networks, International Journal of Scientific and Engineering Research **5** (2014).
[5] e. a. M. Abadi, Tensorflow: A system for large-scale machine learning, in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* (2016) pp. 265–283.
[6] A. Agarap, Deep learning using rectified linear units (relu), (2019).
[7] I. Shafkat, Intuitively understanding convolutions for deep learning (2018).
[8] S. Narayan, The generalized sigmoid activation function: Competitive supervised learning, (1997).
[9] J. Redmonand A. Farhadi, Yolo9000: Better, faster, stronger, arXiv preprint arXiv:1612.08242 (2016).
[10] W. Liuand e. a. D. Anguelov, Ssd: Single shot multibox detector, (2016).
[11] T. Linand M. M. et al., Microsoft coco: Common objects in context, (2014).
[12] P. Shansung, Yolov2 to detect your own objects using darkflow (2018).
[13] Trieu, Darkflow github repository (2018).
[14] F. Shaikh, Understanding and building an object detection model from scratch in python (2018).
[15] B. Cheung, Yolo for real-time food detection (2018).
[16] Y. Agarwal, Create your first image recognition classifier using cnn, keras and tensorflow backend (2018).
[17] N. Tijtgat, How to train yolov2 to detect custom objects (2017).
[18] C. Bishop, *Machine Learning and Pattern Recognition* (Heidelberg Springer, 2006).
[19] H. Muresanand M. Oltean, Fruit recognition from images using deep learning, U. Sapientiae, Informatica **10**, 26 (2018).