Creating a Dataset and Models Based on Convolutional Neural Networks to Improve Fruit Classification

Mihai-Dimitrie Minuț
Faculty of Computer Science
"Alexandru Ioan Cuza" University
Iasi, Romania
mihai.minut@info.uaic.ro

Adrian Iftene
Faculty of Computer Science
"Alexandru Ioan Cuza" University
Iasi, Romania
adiftene@info.uaic.ro

Abstract— **We are often curious to find out what fruit is found in a tree when we are walking through parks or a botanical garden. Fruit classification is a complex activity that depends on the datasets used and the deep learning methods applied to them. This paper aims to present the steps by which we created a dataset with 262 fruits, and then how we created several models based on it. Evaluation and testing have shown us that the resource and models created can be useful to those who want to create applications that recognize fruits.**

Keywords-fruits dataset; model training pipelines; convolutional neural networks; Alexnets

I. INTRODUCTION

The field of artificial intelligence (AI) has evolved a lot in the last years and is still under development which further proves the utility of additional work and research in it. Deep learning and convolutional neural networks are a prime example of the above statement [1] due to the popularity they have in the field of computer vision.

Recently, the popular subjects in the applications for Botanical Gardens are related to fruit and plant recognizing and sometimes to birds and insects recognizing. The current paper presents the elaborated algorithms and methods that help in the development of such applications, regarding the Botanical Garden. Also, it maintains a general perspective on the problems that appear along the way so the results may be useful in as many circumstances as possible: mobile, web or desktop applications.

After Section 2 with review of similar work from domain, the Section 3 is related to the building of a very large dataset of fruits with as many specimens. The dataset should respect the norms and regulations that other public and online datasets respect and should contain information and data of high quality. Next part will be related to building an algorithm that will take as input an image of a fruit and will produce as output a prediction of what fruit the image contains. The aim for the algorithm is to have as much accuracy as possible and to be able to be utilized in other applications that require the feature of fruit recognition. The evaluation section brings into light the strengths and weaknesses of the components mentioned above; multiple statistics and statements, that results from them.

II. STATE OF THE ART

In recent years, neural networks and deep learning algorithms have been used for fruit recognition. In [2], the authors used a method for recognizing and counting fruits. They targeted peppers with fruits of complex shapes and varying colors. The aim of the application is to locate and count green and red pepper fruits. In [3], the authors adapt a faster region-based convolutional network, with the aim to create a neural network that would be used by autonomous robots that can harvest fruits.

The paper [4] shows a network trained to recognize fruits in an orchard. This is a particularly difficult task because the images are taken outside, there is a lot of variance in luminosity, fruit size, clustering, and viewpoint. Related to the automatic harvest of fruits, [5] presents a method of detecting ripe strawberries and apples from orchards. In [6] the authors made a state of the art on methods for harvesting with the aid of robots. In [7], the authors present a fruit dataset for intelligent harvesting. In [8], the authors used backpropagation neural networks to detect apples in order to predict the yield for the upcoming season. In papers [9-13], we can see different approaches to detecting fruits based on color, shape and texture.

In [14], the authors introduce a new, high-quality, dataset of images containing fruits. The dataset was named Fruits-360 and contains 90,483 images of 131 fruits and vegetables in various stages of their life that is provided under the MIT License. The number of labels when compared to a total amount of existing distinct popular classes of fruits [15-18] covers between 30% and 40% of them. While the coverage of distinct fruits seems small, the branching within each distinct species varieties is presented under different labels. A good example would be the apple species, amounting to 13 different subspecies which also include 8 different subspecies types. Based on the Fruits-360 dataset, [19] proposed to use convolution neural networks in building the model and also used ResNet to get the results of image classification.

A review on fruit recognition and feature evaluation using CNN is made in [20] and an intelligent fruit recognition system using deep learning is presented in [21].

III. THE DATASET

For an application that classifies plants and parts of them, the release order of the categories needs to be influenced by how likely the user is to access that category. Most popular or easily found plant categories need to be implemented first. From the pool of common plants or plant parts a regular encounter for humans are fruit, which are usually found through a lot of mediums: any natural habitat, at markets, or

even at home. This makes them one of the better candidates for the first label to have a dataset and an associated model.

A. Labels

From the various sources of fruit labels that can be found on online documentation websites [15-18], the application extracts raw fruit label names through parsing domain specific fruit lists. Adapting them to the specifications implies reformatting their string into a simplified form that simplifies parsing and filtering syntactic duplicates.

A single fruit can have multiple names, including: *regional names, popular names, scientific names,* etc. By storing lists for each fruit label of similar names a duplicate filter can be applied, which removes semantic duplicates. To broaden the search of data for the dataset, each label can be augmented with special keywords which refer to the colors, stages of life or special environments that are associated with each fruit, etc. After the duplicate elimination process the label list was left with 350 labels out of the 650 of the original combined list. For example, for banana, the list is [*"banana", "red banana", "green banana", "yellow banana", "musa sapientum", "musa acuminata"*].

B. Search Engines

The original source that will provide the fruit dataset with data will be the public domain represented through the online image search engines. Each of the selected image search engines, namely "Yahoo", "Google", "GettyImages", "Bing", "Shutterstock" and "Unspash", have a way of querying through images by using related keywords which makes in turn a perfect candidate for the augmented label list. Data that is returned by the search engines, which follows the HTML5 standard, can be parsed and the original images can be found and downloaded, the consequence being the volatility of data found on the Internet (broken resource links, corrupted files, different data formats etc.).

The legal limitations of data obtained using this method is that the resulting dataset must also be declared as being part of the public domain, which in turn makes the usage of models, trained on the data, in commercial projects reside in a gray area, depending on the country laws.

C. Web Crawler

Obtaining data from each of the specified search engine implies building multiple web crawlers hence the format in which the data is presented is not under the same specifications everywhere. The problem of importing data this way consists in finding the way to the image resources deeply nested within the data given by the search engines and then downloading it to a local storage device.

One example of a web crawling algorithm can be found in the **Algorithm 1**, where the process from importing image data from Google is described. To further improve the process of importing data each web crawler will have a unique thread designated which will improve the speed of the imports by treating each engine separately.

Algorithm 1 Import Google Image Data

```
Open chrome webdriver
Access the google image search page while avoiding the user agreement
Access the google search bar by HTML5 element where name = "q"
Type the current fruit label string and press ENTER and wait 2 sec
While not enough images are found and the current search still has images
    items ← all elements with id = "islmp"
    image batch ← all sub elements from items that have the tag "img"
    src batch ← all values of "src" attributes from image batch
    valid srcs ← all valid and accessible links from src batch
    final srcs ← all download links from valid srcs that are not in multithread safe set
    for each element from final srcs
        try to download the image from element under global counter.jpg name
        if any download error appears
            skip the current element
        else
            increase the local image counter
            increase the global image counter
            add the element to the multithread safe set
    if no image data was fetched from last batch
        try to press the button "load more images" that has class name = "mye4qd"
        ignore errors that are thrown
    scroll down in the webdriver window
```

D. Automatic Image Filtering

Within the imported data there are certain elements that must be eliminated, hence they can hurt the process of training a machine learning model or even impact the quality of the final product. Duplicate images or duplicate information can be avoided by storing the image resource links at import time and verifying to not import from the same source twice. The duplicate images that come from different sources can have their data hashed after import, to increase the speed of processing, and then compared in pairs of two to remove the duplicates. Most of the data that came from the predetermined search engines is under the "jpeg" standard, and as a result this was the target format chosen for the dataset, the other ones being removed.

Because of the nature of the data that can come from the Internet, around 1% of the imported data got corrupted during the import phase, and it was eliminated by simply using a script to check all the image data integrity.

Figure 1. Too Wide Acerola Image.

Knowing that the image data will be used to train convolutional neural networks, certain combinations of image width and height can be considered inadequate hence

the loss of data on resize time. Images that have either a very big or small ratio between the width and height (see Figure 1) can be eliminated because of the ulterior reasoning.

To be able to determine the smallest recognizable fruit image size, an image was resized to different dimensions to check where the threshold at which an image loses the minimum required information is (see Figure 2). As a result the images that had less than 64 × 64 pixels worth of information were eliminated.

Figure 2. Resizing Acerola Image from X to 256.

E. Manual Image Filtering

After the process of automatic filtering was done the average image information has risen but the images still were full of elements that were not related to the task (see Figure 3).

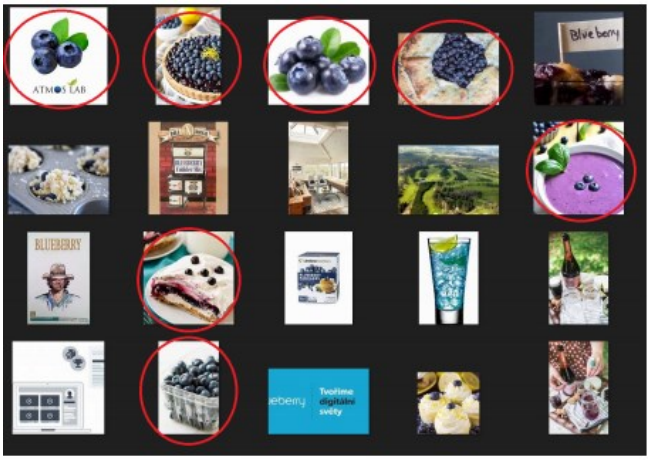

Figure 3. Blueberry Label – Images Containing Blueberries.

One of the available options for solving this problem was using an AI that was already trained to distinguish fruit images from the rest. Because using a fruit recognition AI to build a fruit recognition AI was defeating the purpose of the project, this approach was discontinued.

The other option, which was also the chosen one, is to establish a team of people, which will choose suitable images, and a filtering paradigm, thus establishing a quality standard for the images. The paradigm has suffered modification through the process, and in the end has come to the following schema:

I. Gather Information

To be able to select viable images of a certain label certain information about the fruit and its life is required; one should be able to answer the following questions, in no particular order of importance, in order to start filtering the image data:

- *What is the habitat or habitats that the fruit might grow in?*
- *What are the colors and shapes the fruit might have in any stage of its life?*
- *Is the fruit usually found in groups?*
- *How does the interior of the fruit or its slices look like in any stage of its life?*

To acquire this kind knowledge any method can be followed, the most used ones including searching information and images on the Internet or looking up botanic books.

II. Select Fitting Images

The second step is to start iterating in any order utilizing any type of visualizer for the images and selecting items that respect the next criteria, until the thousand target or the end of the label data is reached. If any of the items falls in a gray area, it should be saved separately in case there is a need for more data at the end of sorting. The most popular applications used for iterating and visualizing being in this case Windows Explorer and respectively Paint.

II. 1. Criteria of MUST

All images that are to be selected have the requirement to follow the following properties:

- have at least 50% of the pixels occupied by the respective fruit's data;
- (OR) contain at least the entirety of at least one fruit from the label;
- (OR) contain at least the entirety of at least one slice of the fruit from the label.

II. 2. Criteria of MAY

All images that are to be selected can have one or multiple of the following properties:

- contain the fruit in any stage of its life;
- contain any type/color/form of the label;
- have the background representing the natural habitat of the fruit counted towards the 50% fruit information;
- contain other items other than the fruit and its related habitat;
- be formed through the combination of multiple smaller images, if the data is scarce for the current label; these falling usually under the gray area category.

II. 3. Criteria of SHOULD NOT

All images that are to be selected should not have any of the following properties:

- contain any other fruits or fruit parts than the ones coming from the label;
- contain the fruit represented by a drawing and, not a natural instance of the fruit;

- have obvious filters applied to the image that alter the fruit information.

III. Corner Cases

There may be situations where an individual cannot be sure what to do with some items while also not having enough data to hit the 1,000 mark set on the label. For these peculiar 34 situations all the team participated and voted on what to do with them.

By the end the team was composed of six (five males and one female) members, with ages between 16 and 22 years. Each of them having contributions ranging from small to large to both the selection paradigm and the resulting dataset.

F. Resulted Dataset

After filtering and going through all the label removals and merges (when 2 labels had identical fruits), the final dataset came to have a total of 262 distinct labels from the 293 resulted from the original importing.

The resulted dataset now amounts to 225,640 images which translates to approximately 2.7 GB of data. The dimension of the dataset decreased by a magnitude of 9, which is very realistic considering the target was to reduce the 10,000 images to 1,000 images per label and also the removals.

The total amount of time for producing the dataset was around 7-8 months, from which 2 were allocated for importing the data, 2-3 for manual filtering and the other 3 for planning, producing the software needed and also researching the topic. The final state of the dataset quality can be seen in Figure 4.

Figure 4. Blueberry Images after Manual Filtering.

G. Statistics

Comparing the statistics made on the final dataset (Table I.) and the ones made before manual filtering (Table II.), while also considering the decrease in the order of magnitude it can be clearly stated that the average number of images per label remained the same, but the median went up by about 33% and the Standard Deviation halved.

This result proves that the balance of the dataset has risen along with the number of labels that respect the target number of images. The information loss coming from the filtering can be estimated at about 14%, but the increase in quality per pixel makes it justifiable.

TABLE I. FINAL DATASET

	Average	**Median**	**Standard deviation**
Images	861	1007	276
Images Width	213	209	19
Images Height	262	255	30
Missing Images	580	567	258

TABLE II. INITIAL DATASET

	Average	**Median**	**Standard deviation**
Images	8,942	7,517	5,736
Images Width	258	249	39
Images Height	304	304	38
Missing Images	5,953	6,010	2,248

H. Problems

The current dataset is usable by machine learning algorithms but the amount of images per label, considering the number of labels, is still very low, implying that the resulting models will be more inefficient when separating fruit types. Considering the provided statistics (Table I and II), the image information is not equally distributed to each of the labels, resulting in an imbalanced dataset. Some images, from the same label, are very similar and will probably cause overfitting problems for the machine learning algorithms (see Figure 5).

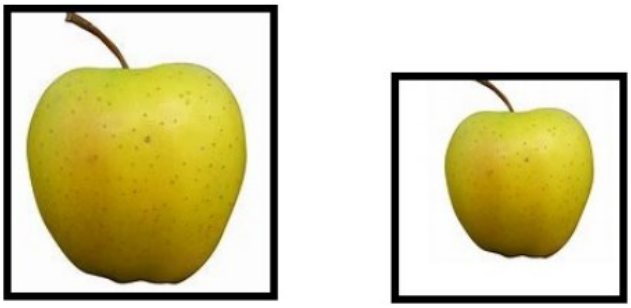

Figure 5. Same Apple – Similar but Different Images.

IV. FRUIT IMAGE CLASSIFIER

Given the state of the art for image recognition, and also considering the project goal of obtaining the best feasible model, the decision to focus solely on building and optimizing pipelines for training Convolutional Neural Networks was the safest one.

A. Data Splitting

The dataset had the images sorted by labels, each with its respective directory named after the fruit name. In order to split the data into training, validation and testing a script was created which loaded all the relative image file paths for each fruit, then split the paths into categories respecting the ratio of 70% for training, 20% for validation and 10% for testing.

B. Image Dimension Standard

CNN models generally accept as input only a set shape for the information, and as a consequence forming a standard

for dimensions, which will also improve the final statistics. In order to split the data into training, validation and testing a script was created which loaded all the relative image file paths for each fruit, then split the paths into categories respecting the ratio of 70% for training, 20% for validation and 10% for testing.

The standard that will be used is based on images being resized to a size at which the loss of information will be minimal from resizing. Considering the information from the statistics table of the dataset, the least loss of data comes from the middle point of 213 × 262 pixels, but this does not account for the data resizing throughout the pooling layers of the convolutional model.

The calculation for the best standard took the above statement to the core, and as a result the method for calculating the loss of information is:

```
f(dimension) = |average_information –
                dimension × 16|,
```

where `dimension` ∈ N*. Optimizing the above formula for minimum data loss gives the standard of 13 × 16, 26 × 32, 52 × 64, 104 × 128 and 208 × 256 dimension ratio for resizing the images.

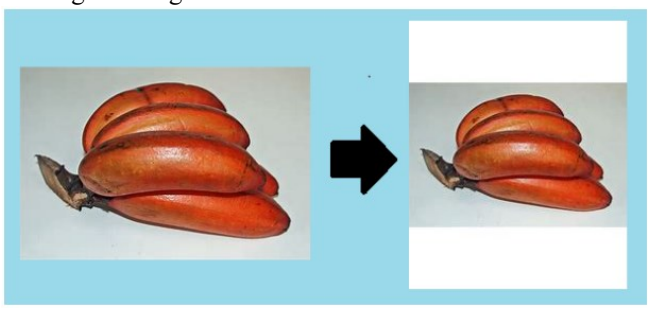

Figure 6. Banana Image Resized.

The process for resizing an individual image (Figure 6) to a set sub part of the standard will be:

- creating a new blank image with the standard size,
- resize the fruit image, equally on width and height, to fit into the new blank image,
- insert the resized image at the center of the new one and save the result.

C. *Efficient Storage*

The problems that occurred during the first iterations of loading data were loading time and loading spaces.

To solve the above mentioned problems all the image files were loaded into an array then serialized into one or multiple files, thus reducing the IO operations and also the amount of data that needed to be loaded.

TABLE III. EFFICIENT LOAD

Dimensions	Load all data	Efficient load all data	Efficient load one pack	Train packs	Valid. packs	Test packs	Size HDD	Size RAM
13×16	10s	1s	1s	1	1	1	81Mb	200Mb
26×32	31s	4.7s	4.7s	1	1	1	300Mb	1Gb
52×64	1.5 min	15s	15s	1	1	1	1Gb	3.5Gb
104×108	7 min	2min	27s	3	1	1	1.1Gb	4Gb
208×256	30 min	10min	36s	12	4	2	1.2Gb	4Gb

The end result (Table III. Efficient Load) features much faster data loading and, for the higher dimensions of the standard, the data was segmented into multiple packs, which occupy up to 4GB in RAM size, which will be preloaded one by one during the training time on a separate thread, thus reducing the IO loading data blocking process.

D. *Initial feasibility tests*

After building the dataset and the loading mechanism the next step is testing simple model architectures to see how they behave using the data.

TABLE IV. SIMPLE MODEL

Units	Layer	Kernel Size	Activation	Layer Active Condition
8 filters	Convolutional2D	3×3	Relu	padding, 208×256×3
-	MaxPooling2D	2×2	-	padding, 208×256×3
16 filters	Convolutional2D	3×3	Relu	padding, >=104×128×3
-	MaxPooling2D	2×2	-	padding, >=104×128×3
32 filters	Convolutional2D	3×3	Relu	padding, >=52×64×3
-	MaxPooling2D	2×2	-	padding, >=52×64×3
64 filters	Convolutional2D	3×3	Relu	padding, >=26×32×3
-	MaxPooling2D	2×2	-	padding, >=26×32×3
128 filters	Convolutional2D	3×3	Relu	-
-	MaxPooling2D	2×2	-	-
256 filters	Convolutional2D	3×3	Relu	-
-	Flatten	-	-	-
128	Dense	-	Relu	-
262	Dense	-	Softmax	-

The model presented in Table IV features a traditional CNN that is adaptive to each of the manufactured image standards. The algorithm for optimization that will be used is Adam and the method for computing the loss will be Categorical Cross entropy. Considering the output of the final convolutional layer being 4 × 2 × 3, the number of parameters were set to be small in the dense layers, so the model produces results faster.

The filter sizes and their number were kept low while also respecting the rule of increasing the filter amount gradually so the training process is much more efficient [22].

TABLE V. SIMPLE MODEL – TRAINING TESTS (SMTT)

Index	Input dimension	Epoch time	Epochs number	Train accuracy	Validation accuracy
1	13×16	12s	1,000	40%	22%
2	26×32	28s	1,000	60%	29.5%
3	26×32	28s	3,000	99.4%	~
4	52×64	1min	1,000	65%	33.33%
5	104×128	4min	250	~	~
6	208×256	20min	50	~	~

These choices were natural for the problem presented, separating images coming from a large variety of labels in the simplest way to quickly produce statistics. The test results for the simple model (Table V. SMTT) provide

crucial information about training time, model accuracy and the dataset.

The tests labeled 1, 2, 4 prove that training models on the 13 × 16, 26 × 32 and 52 × 64 dimensions standards is feasible time wise, while the ones marked with 5 and 6 clearly prove that training on 104 × 128 and 208 × 256 standards consume time on a much larger scale and, as a result, experiments on them will be discontinued.

The test labeled as **3** proves that even a lower dimension standard, such as 26 × 32, of the dataset contains enough information for a model to completely overfitting and separate the fruit images, which proves the dataset usability.

The resulted models also come with very good validation accuracy scores considering the amount of 262 distinct labels and the scarce amount of training data.

E. Advanced Models

In order to obtain better models, options such as advanced CNN techniques and building on already successful architectures will be explored. The input values of the pixels were divided by 255 to translate them to the [0, 1] domain, thus increasing training efficiency by using small values for weights [22]. Different optimizers were tested on varying learning rates but the best results were given by Adam and Adamax.

TABLE VI. FRUITS-360 PAPER TRAINING PIPELINE + SOME ADDITIONS (FPTP)

Units	Layer	Kernel size	Activation	Additional info
-	random_saturation	-	-	*flip_left_right
-	random_hue	-	-	*flip_up_down
-	hsv_and_grayscale	-	-	*removed
-	rescale	-	-	scaling the pixels by /255
16 filters	Convolutional2D	5×5	Relu	with padding
-	MaxPooling2D	2×2	-	-
32 filters	Convolutional2D	5×5	Relu	with padding
-	MaxPooling2D	2×2	-	-
64 filters	Convolutional2D	5×5	Relu	with padding
-	MaxPooling2D	2×2	-	-
128 filters	Convolutional2D	5×5	Relu	with padding
-	MaxPooling2D	2×2	-	*only in the improved version
256 filters	Convolutional2D	3×3	Relu	with padding *only in the improved version
-	Flatten	-	-	-
1024	Dense	-	Relu	-
-	Dropout	-	-	20% rate
252	Dense	-	Relu	*improved version has 1024 units
-	Dropout	-	-	20% rate
262	Dense	-	Softmax	output

The image formats, used for the input, that proved themselves the best were RGB and the combination of RGB with HSV and Grayscale (stacked on top of each other).

Augmentation with image flipping, brightness spots and image rotation along with dropout layers in the dense layers proved to decrease the overfitting factor during training while also increasing the accuracy ceiling. Layer Regularization was tested and due to the big amount of time added to training was dropped.

The paper that comes with the "Fruit-360" [23] dataset, namely "Fruit Recognition from images using deep learning" [14], provides a pre build architecture. By adapting it to the current problem (Table VI. FPTP) and training a model, the end result features a 53.86% validation accuracy along with a 75% training accuracy (see Figure 7). Other tested base architectures were the VGG16 [24], VGG19 [24], Lenet5 [25] and AlexNet [26]. The end results for each adapted architecture features models with validation accuracy above 50%, the best one being based on AlexNet with a 55% validation accuracy.

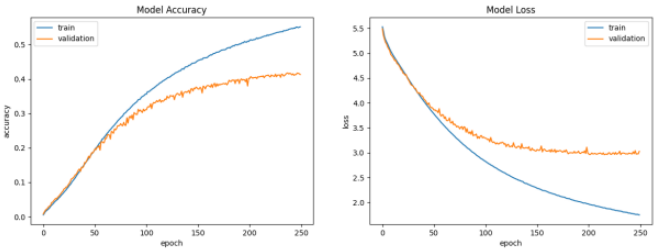

Figure 7. FPTP Model – Training Session.

F. Final Training Pipeline

Having done the advanced techniques and architectures tests, the tests for final training pipelines could start. The base for the model structure will be the previously built AlexNet structure (Table VII. FTP), hence it proven as the best structure tested.

TABLE VII. FINAL TRAINING PIPELINE (FTP)

Units	Layer	Kernel size	Activation	Additional info
-	flip_left_right	-	-	-
-	flip_up_down	-	-	-
-	Convert to gray + HSV + RGB	-	-	*only second model
-	rescale	-	-	scaling the pixels by /255
64	Convolutional2D	11×11	Relu	padding
-	MaxPooling2D	2×2	-	-
128	Convolutional2D	5×5	Relu	padding
-	MaxPooling2D	2×2	-	-
256	Convolutional2D	3×3	Relu	-
256	Convolutional2D	3×3	Relu	-
256	Convolutional2D	3×3	Relu	-
-	MaxPooling2D	2×2	-	-
-	Flatten	-	-	-
-	Dropout	-	-	50% rate
1024	Dense	-	Relu	-
-	Dropout	-	-	50% rate
1024	Dense	-	Relu	-
262	Dense	-	Softmax	output

The data will be augmented by flipping it with a chance horizontally and vertically, as it proved the most effective. As an experiment two models will be trained: one with the standard RGB as input (first FTP model) and the other with

the RGB + HSV + Grayscale (second FTP model). The convolutional layers will be organized to resemble the ones from the AlexNet structure. Different kernel sizes have been tested but the end results is presented in the final pipeline (FTP). The amount of filters is as high as it could be, but still trainable in a reasonable amount of time, the upper limit being established in one week.

The output of the final convolution layer will be as 4 × 2 × *color channels*. This is due to the combination of pooling layers with convolution, with or without padding added to them (FTP). The final fully connected layers are in number of 3, one from which is the output, because of the tests conducted before with FTPT. The final count for the trainable parameters is around 5 million.

The training session has been done with a learning rate of 25×10^{-5} in batches of 256 images. The RGB model took around 130 epochs to reach the ceiling for accuracy (57% validation accuracy, 75% training accuracy) and the RGB + Grayscale + HSV model peaked in 150 epochs (56% validation accuracy, 66% training accuracy) (see Figures 8 and 9).

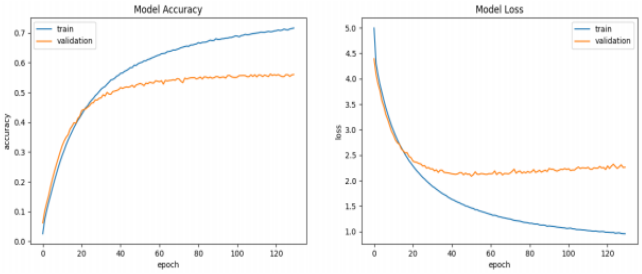

Figure 8. FTP – First Model – Training Session.

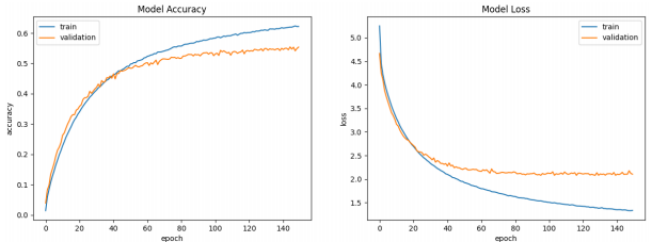

Figure 9. FTP – Second Model – Training Session.

The models produced by the final training pipelines were tested with the 10% remaining data of the dataset that was not used so far (**TDR**).

TABLE VIII. FINAL MODELS – TEST DATA RESULTS (TDR)

Model	Top 1 prediction accuracy	Top 5 prediction accuracy	Top 10 prediction accuracy
13×16 the best test model obtained	42.47%	58.57%	65.88%
52×64 RGB	59.15%	80.4%	86.66%
52×62 Grayscale + RGB + HSV	58%	79.24%	85.69%

The pipeline for testing was similar to the one for training. Test images were also a subject of augmentation, to further increase the pool of images and also give a better and stable look at the end accuracy.

Since the testing pipeline was not deterministic, each image was put through it 1,000 times for the statistics to stabilize.

The output of the model was represented by a list of predictions for each label in the dataset. The accuracy, as a result, was measured by checking if the correct label was present in the top 1, 5 and 10 predictions given by the model (see Figure 10).

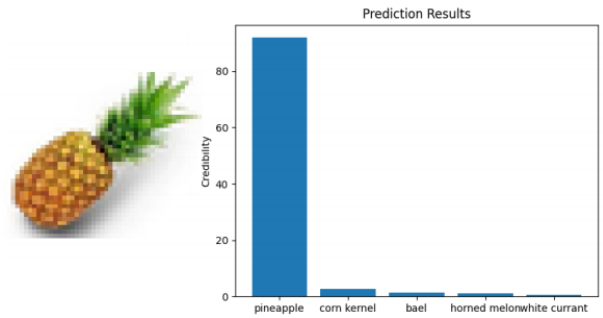

Figure 10. Pineapple Image – Top 5 Predictions.

G. *Problems*

The decision to train on the imbalanced dataset and having labels with under 100 specimens and others with over 1,000 definitely has negatively impacted the end results. In further experiments, labels with small numbers of images could be eliminated or integrated in similar fruit families.

One big problem, encountered while testing the feasibility of the simple models, is that loading a lot of data at once from the storage device can take a long time and in turn make training a model very slow. The solution concluded in storing the dataset in a much streamlined way, which allowed for faster loading and no time wasted on changing the structure of information.

One problem discovered during the final stages of the project, specifically when building the prediction pipeline, is that some of the ”berry” fruit types tend to be very similar; to the point that even the human eye has trouble distinguishing between them (see Figure 11).

Figure 11. Hackberry vs Blueberry.

Choosing the perfect model architecture for a problem that falls under the domain of neural networks is very difficult and becomes even more complex. The scarcity of similar tasks faced by CNN models also adds more complexity to choosing the right structure.

V. CONCLUSIONS

The project has started with the goal of having a dataset for plants on which to build CNN models for classification. In the end, the project managed to generate the "Fruits-262" dataset [27] from fruit images found on the public domain, which also led to some averagely good CNN models.

For building the better final models, which have around 58% test accuracy, a lot of initial testing and exploring have been done. Data augmentation and the Dropout Layers pushed the models to higher plateau of accuracy, and together, with the structure of AlexNets and a good amount of trainable weights have obtained the peak results. While testing the final models by using as input individual fruit images, another problem arose, one which involves various species of berries being hard to distinguish. It turns out that some berries are hard to compare when they are allowed to be in peculiar positions.

Having images with a high degree of similarity, under the same label type, leads to a higher end model accuracy since they can be treated as approximate duplicates but negatively impacts how the model performs in real scenarios. A further iteration of the dataset could remove this problem by using the said instances of images and transform them with some method of augmentation.

ACKNOWLEDGMENT

This work was support by project REVERT (taRgeted thErapy for adVanced colorEctal canceR paTients), Grant Agreement number: 848098, H2020-SC1-BHC-2018-2020/ H2020-SC1-2019-Two-Stage-RTD.

REFERENCES

[1] A. Moghariya, "Why CNNs are used more for computer vision tasks than other tasks?" Quora.com, 2017.

[2] Y. Song, C. Glasbey, G. Horgan, G. Polder, J. Dieleman, and G. van der Heijden, "Automatic fruit recognition and counting from multiple images," Biosystems Engineering, vol. 118, 2014, pp. 203-215.

[3] I. Sa, Z. Ge, F. Dayoub, B. Upcroft, T. Perez, and C. McCool, "Deepfruits: A fruit detection system using deep neural networks," Sensors, vol. 16, issue 8, 2016.

[4] S. Bargoti, and J. Underwood, "Deep fruit detection in orchards," In 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 3626-3633.

[5] S. Puttemans, Y. Vanbrabant, L. Tits, and T. Goedeme´, "Automated visual fruit detection for harvest estimation and robotic harvesting," In 2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA), 2016, pp. 1-6.

[6] K. Kapach, E. Barnea, R. Mairon, Y., Edan, and O. Ben-Shahar, "Computer vision for fruit harvesting robots: state of the art and challenges ahead," International Journal Computer Vision Robot, vol. 3, issue 1/2, 2012, pp. 4-34.

[7] H. Altaheri, M. Alsulaiman, G. Muhammad, S.A. Amin, M.A. Bencherif, and M. A., Amine, "Date Fruit Dataset for Intelligent Harvesting," Data in Brief, vol. 26, 2019, 104514. 10.1016/ j.dib.2019.104514.

[8] H. Cheng, L. Damerow, Y. Sun, and M. Blanke, "Early yield prediction using image analysis of apple fruit and tree canopy features with neural networks," Journal of Imaging, vol. 3, issue 1, 2017.

[9] A. Selvaraj, N. Shebiah, S. Nidhyananthan, and L. Ganesan, "Fruit recognition using color and texture features," Journal of Emerging Trends in Computing and Information Sciences, vol. 1, 2010, pp. 90-94.

[10] J. Xiong, Z. Liu, R. Lin, R. Bu, Z. He, Z. Yang, and C. Liang, "Green grape detection and picking-point calculation in a night-time natural environment using a charge-coupled device (ccd) vision sensor with artificial illumination," Sensors vol. 18, issue 4, 2018.

[11] H. Zawbaa, M. Abbass, M., Hazman, and A. E. Hassanien, "Automatic fruit image recognition system based on shape and color features," Communications in Computer and Information Science, vol. 488, 2014, pp. 278-290.

[12] P. Ninawe, and M.S. Pandey, "A completion on fruit recognition system using k-nearest neighbors' algorithm," In International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) vol. 3, 2014.

[13] D. Li, H. Zhao, X. Zhao, Q. Gao, and L. Xu, "Cucumber detection based on texture and color in greenhouse," International Journal of Pattern Recognition and Artificial Intelligence, vol. 31, 2017.

[14] H. Muresan, and M. Oltean, "Fruit recognition from images using deep learning," Acta Univ. Sapientiae, Informatica, vol. 10, issue 1, 2018, pp. 26-42.

[15] Half Your Plate, Fruits, 2021 https://www.halfyourplate.ca/fruits-and-veggies/fruits-a-z/.

[16] Wikipedia, List of All Fruits, 2021 https://simple.wikipedia.org/wiki/ List_of_fruits.

[17] List Challenges, Every Fruit in the World, 2021 https://www. listchallenges.com/coolfruits.

[18] Fruitsinfo, List of All Fruits, 2021 https://www.fruitsinfo.com/fruit-list.php.

[19] P. Reddy, S. Ramasubbareddy, D. Saidulu, and K. Govinda, "Fruit Recognition Using Deep Learning," Innovations in Computer Science and Engineering, LNCS 171, 2021 pp. 53-62, 10.1007/978-981-33-4543-0_7.

[20] D.N.V.S.L.S. Indira, J. Goddu, B. Indraja, V. M. L. Challa, and B. Manasa, "A review on fruit recognition and feature evaluation using CNN," Materials Today: Proceedings, 2021.

[21] H.H.C. Nguyen, A. T. Luong, T. H., Trinh, P.H. Ho, P. Meesad, and T.T. Nguyen, "Intelligent Fruit Recognition System Using Deep Learning," In: Meesad P., Sodsee D.S., Jitsakul W., Tangwannawit S. (eds) Recent Advances in Information and Communication Technology, IC2IT 2021, Lecture Notes in Networks and Systems, vol 251. Springer, Cham, 2021, https://doi.org/10.1007/978-3-030-79757-7_2.

[22] D. Stutz, "Understanding Convolutional Neural Networks," Fakultat fur Mathematik, Informatik und Naturwissenschaften Lehrund Forschungsgebiet Informatik VIII, Computer Vision, 2014.

[23] M. Oltean, "Fruits 360 - A dataset with 90380 images of 131 fruits and vegetables," 2017, https://www.kaggle.com/moltean/fruits/ metadata.

[24] K. Simonyan, and A. Zisserman, "Very Deep Convolutional Networks for Large Scale Image Recognition," 2015 arXiv:1409.1556v6

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," Proceedings of the IEEE, vol. 86, no. 11, 1998, pp. 2278-2324, doi: 10.1109/5.726791.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Communications of the ACM, vol. 60, issue 6, 2017, pp. 84-90, https://doi.org/10.1145/3065386.

[27] M.D. Minuț, "Fruits-262 - A dataset containing 225,640 images of 262 different fruits," 2021, https://www.kaggle.com/dataset/ f9472b258bbdab0dbc8cc773ad8c78a2fa1b997fa0cd88a476f184b78b93338c.